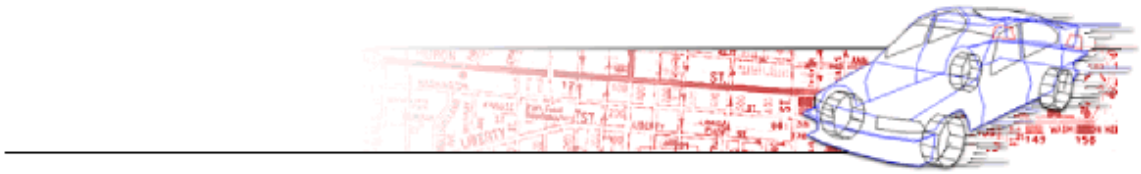


CarSim Educational

User Manual

VERSION 4.5

Mechanical Simulation Corporation
January 2000

NOTICE

This manual describes the CarSim Educational software.

© 1996 – 2000, Mechanical Simulation Corporation. All Rights Reserved.

Mechanical Simulation Corporation
709 W. Huron, Ann Arbor, MI 48103
Phone: (734) 668-2930
FAX: (734) 668-2877
http://www.trucksim.com

Table of Contents

1. Introduction 9

About This Manual 9
What CarSimEd Does 9
CarSimEd Model Features 11
How CarSimEd Works 13
What CarSimEd Does Not Do 15
Notational Conventions in This Manual 16

2. CarSimEd Installation 17

Computer Requirements 17
Installation of CarSimEd 18
Linking to SIMULINK and MATLAB 18

3. Database Organization 21

Introduction to the CarSimEd Database 21
Data Screens 22
Making New Data Sets 24
On-Line Help 24
A Map of the CarSimEd Libraries and Screens 24
How CarSimEd Communicates with SIMULINK 28

4. The Basics of Using CarSimEd 30

About the Runs Screen 30
Getting to a Runs Screen 31
Making a New Run (Stand-Alone) 33
Replacing a Run (Stand-Alone) 34
Making the First SIMULINK Run 35
Making Additional SIMULINK Runs 36
Viewing an Animation 36
Viewing a Single Pre-Defined Plot 37
Viewing Several Pre-Defined Plots 38
Overlaying Plots for Multiple Runs 38
Defining Plots Interactively 39
Viewing All Model Parameters and Inputs 40
Printing a Data Set 40
Making a New Vehicle Data Set 41
Modifying an Existing Vehicle Description 42
Switching Between CarSimEd and SIMULINK Inputs 43
Going Directly to Any CarSimEd Library 44
Locking Your Data 45
Deleting Data Sets 48
Making Five or More Plots for a Single Run 48

Overlaying Plots for Multiple Runs 49
Making Many Plots in Batch Mode 50
Setting Up Multiple Runs 50
Making New Plot Descriptions 52
Linking to Different Libraries 53

5. The Solver Programs 55

Overview of Program Operation (Stand-Alone) 55
How a Simulation Run Is Made in SIMULINK 56
File Types 59

6. The Animator 63

Updates 63
Overview of Operation 63
Reference Frames 64
Files 65
Units 66
File Menu 67
Edit Menu 69
Animation Menu 70
Coordinates Menu 71
Options Menu 73
Help Menu 75
Pop-Up Menu 76
Time Control Slider 76
Testing Animator Data Sets 77

7. The Plotter 80

Updates 80
Overview of Operation 80
Batch and Interactive Operation 81
Zooming 82
Tool Bar 83
Printing Plots 84
File Menu 86
Edit Menu 88
Format Menu 90
Data Menu 96
View Menu 105
Windows Menu 107
Help Menu 109

8. Design of CarSimEd Data Screens 110

The Ribbon Bar 110
Data Links 115

Tabular Data 117
File Menu 119
Edit Menu 122
Text Menu 123
Page Menu 124
Tools Menu 125

9. Alphabetical Library Reference 127

Conventions in This Chapter 127
Animator: Camera Setup 129
Animator: Groups 133
Animator: Reference Frames 136
Animator: Shapes 139
Animator: Wheels 142
Calculator 143
CarSimEd Startup 150
Computation Parameters 155
Generic 2D Table 157
Generic Data Group 159
Generic Table 161
Input: Braking 162
Input: Road Profile 164
Input: Steering Wheel Angle 166
Input: Target Path For Closed-Loop Steer Control 167
Input: Throttle Control 171
Input: Wheel Height Above Ground 173
Library Editor 174
Plot Format 176
Plot Setup: Batch 179
Plot Setup: Single 181
Plot Transforms 183
Runs: 2D Ride 187
Runs: 3D Handling 188
Runs: SIMULINK CMEX Version 196
Runs: Suspension Analyses 199
Runs: Batch 201
Suspensions: Independent 204
Suspensions: 5-Link Independent 207
Tires: CarSimEd Model 209
Tires: Cornering Stiffness 212
Tires: Pneumatic Trail 214
Vehicles: Car 215

10. Advanced Topics 218
Adding a New SIMULINK Model 218
Installing CarSimEd in a New Directory 219
Importing Data from Another Copy of CarSimEd 222
Exporting Data to Other Plotting and Analysis Software 223
Changing Plot Formatting 225
Creating Offset Plots 226
Re-Scaling Tabular Data 227
Improving the Appearance of the CarSimEd Screens 228
Continuing a Run 230
Changing the Default Runs Library 232
Changing a Solver Program 232
Changing the Default Text Editor 234

11. Trouble Shooting 236
File System Errors 236
Database (ToolBook) Problems 237
Solver Programs 239
Plotter and Wire-Frame Animator 239

Appendix A — Glossary 241

Appendix B — Vehicle Dynamics Terminology 245
Vectors and Angles 246
Axis Systems and Coordinate Systems 247
Entire vehicle 250
Suspensions and steering 253
Tires and wheels 255
Notes 256

Appendix C — ERD File Format 258
The Header 258
The Data Section 262

Appendix D — Plotter Files and Keywords 264
PLT Batch Control Files 264
Plot Setting Files 265
Plot Transform Files 266
Plot Format Files 266
Preference File Format 269
Text Files 269

Appendix E — Animator Files and Keywords 271
Overview of a PARSFILE 271
Units, Coordinates, and Sign Conventions 274
Camera Settings 274
Reference Frames 275
Parts (Shapes) 277
Wheels 279
Grid and 3D Ground Surface 280
Target Path 282

Appendix F — Model Files and Keywords 283
File Types 283
Viewing Keywords 287
Keywords and Parameters for the 3D Car Model 288
Keywords and Parameters for the 2D Car Model 292
Keywords and Parameters for the 3D Suspension 294

Appendix G — The 3D Car Model 296
Introduction 296
Overview of Factors Affecting Vehicle Behavior 297
Rigid Body Kinematics 298
Masses and inertias 300
Suspension Force Effects 300
Summary of Major Model Variables 302
References 306

Appendix H — The Tire Model 307
Tire/Wheel Kinematics 307
Tire Forces and Moments 312
Low-Speed Exceptions 314
Sequence Of Calculations 316
Sign Conventions 316
References 317

Appendix I — The CarSimEd Steer Controller 318
Optimal Control Theory 318
Application to Vehicle Control 320
References 325

Appendix J — Model Input and Output Variables 326
Types of Output Variables 326
Inputs and Output Variables 328
Creating Lists of Outputs 335

1. Introduction

This chapter begins an overview of the rest of this manual. It then continues with a summary of what CarSim Educational can do and how it works.

Throughout this manual, CarSim Educational is called CarSimEd.

About This Manual

This manual is intended to help you understand and use CarSimEd.

- The first three chapters provide an overview of CarSimEd: its use, installation, and program structure.
- Chapter 4 provides instructions for performing common operations in CarSimEd.
- Chapters 5 through 9 provide reference material. Chapter 9, the largest in the manual, describes the CarSimEd data screens in alphabetical order.
- Chapter 10 provides instructions for performing operations that are more advanced than those covered in Chapter 4.
- Chapter 11 contains tips to help diagnose problems and identify errors.
- The appendices provide technical details such as a glossary, vehicle dynamics terminology, file formats, model parameter definitions, output variable names, and assorted modeling details.

Although we would not discourage you from reading this manual from beginning to end, we recognize that your time is valuable and that you might start by reading only what you think is absolutely necessary. **Please read Chapters 1 through 4**. In particular, Chapter 4 has the essential information about how you use CarSimEd. The majority of user questions from the past few years are covered in this chapter. Other chapters can be read as needed or for reference.

The most recent version of this manual can be obtained (free) from the web site: **`www.trucksim.com/carsimed/index.html`**

What CarSimEd Does

CarSimEd is a software package for simulating and analyzing the behavior of four-wheeled vehicles in response to steering, braking, and acceleration inputs. It produces the same kinds of outputs that might be measured with physical tests involving instrumented vehicles.

CarSimEd Advantages

Detailed mathematical models for simulating automotive vehicle dynamics have been in use for decades. However, simulation with detailed models has too often been considered to be an advanced form of analysis that is only practical for experts in simulation technology and vehicle dynamics.

Here are some reasons that CarSimEd makes simulation practical for students and engineers:

1. It is much easier to use than older vehicle simulation programs. All aspects of the software have a point-and-click graphical user interface.
2. It requires no special hardware. CarSimEd runs on ordinary Windows desktop and laptop PCs.
3. It runs fast. Simulations run faster than real time on any current PC.
4. It includes a database that minimizes the time needed to build a vehicle description and set up run conditions. Vehicles, components, inputs, existing runs — all kinds of data — are accessible with pull-down menus in the database.
5. CarSimEd requires no additional software. It includes a graphical database, solver programs, plotter, animator, and utility programs. (However, CarSimEd is designed to also work with MATLAB®/SIMULINK®, as explained on page 12.)

Due to its ease-of-use and speed, you can run simulated tests far more quickly and easily in CarSimEd than with physical testing. It's easy to simulate vehicles and test conditions that are difficult or impossible to conduct experimentally. After you run a simulation you can inspect and analyze hundreds of variables.

Why Use Simulation?

The reason for using a vehicle simulation program is nearly always to learn something about vehicle behavior. Traditionally, simulations have been used by experts to learn about specific issues, such as:

1. How does a specific design change affect vehicle response in certain test conditions?
2. Why does an existing vehicle behave in an unexpected fashion under certain specified conditions? (And what can be done to change that behavior?)
3. What were the conditions leading to a specific accident or crash? (Can the event be reconstructed?)

CarSimEd is designed to provide accurate simulation with rapid speed and ease-of-use. This means it can be used by students and engineers who are not yet experts in vehicle dynamics, to rapidly gain experience in the subject.

The same factors (speed and ease of use) provide the potential to provide an impact on design decisions much earlier in the design process than has traditionally been possible.

CarSimEd Operation Summary

The basic operation of CarSimEd involves this sequence:

1. Select a vehicle to run. As needed, modify its properties (dimensions, mass, etc.) components (brake system, steering system, shock absorbers, springs, etc.), and possibly the properties of its components. If you use only existing data sets, you can assemble the vehicle with pull-down menus alone.
2. Specify the control inputs for steering, braking, and throttle. (Again, this can be done with pull-down menus if you are using existing data sets.)
3. Run the simulation with a single button click. (As the run proceeds, CarSimEd writes force and motion variables into an output file for later analysis.)
4. Click a button to view an animation of the simulated test to get an overall view of the vehicle behavior.
5. Click another button to view plots and analyze the resultant behavior in more detail.

With practice, you will be able to quickly see how component and vehicle design changes affect critical aspects of behavior such as braking performance, understeer, and transient response.

CarSimEd Model Features

This section provides a short summary of the features of CarSimEd that pertain to vehicle dynamics and its mathematical models.

Models

CarSimEd has several separate simulation programs. Each solves equations of motion numerically for a mathematical model designed to predict an aspect of vehicle behavior. The models are:

1. A comprehensive 3D handling model that computes response time histories for steering, braking, and throttle inputs.
2. A 2D pitch-plane model for simplified ride analyses.
3. A 3D kinematics model (no dynamics or compliance) of a 5-link suspension.

3D Handling Model

The main simulation program in CarSimEd is a handling model with full 3D rigid-body equations for the vehicle sprung and unsprung masses. The major kinematical and compliance effects of the suspensions and steering systems that affect wheel motions are included, and are represented with parameters that correspond to standard vehicle properties that can be measured directly. Details of the linkages and gears in the

suspensions and steering systems are not included, reducing the amount of information needed to obtain accurate simulations.

The model has three forms of input: steering, braking, and throttle. Steering can be specified explicitly as a function of time. Alternatively, a path input can be specified (e.g., a circle) and a built-in driver model will generate the steering wheel angle as needed to try to follow the path. Braking and throttle inputs are both specified as tabular functions of time.

Two solver programs are provided for this model. One is a stand-alone application program with the file extension EXE, and the other is a plug-in for use with MATLAB/SIMULINK with the extension DLL.

2D Ride Model

The ride model uses a subset of the input parameters required for the more detailed 3D model. Information related to roll or yaw motion is ignored. This model can be used to determine how basic ride parameters (wheelbase, mass center, spring and damper rates, etc.) affect the behavior of a car going over an arbitrary road profile.

3D Suspension Model

The suspension model is of a 5-link design, as described in the IAVSD benchmark (M. Hiller and S. Frik, "Five-Link Suspension," from *Multibody Computer Codes in Vehicle System Dynamics*, Supplement to *Vehicle System Dynamics*, vol 22, 1993. pp 254 - 262). The model can simulate short-long-arm (SLA) wishbone suspensions, and also the 5-link rear suspensions used in some Daimler-Benz automobiles.

Outputs

You can view simulation results as wire-frame animations or as plots of output variables. All it takes is a single button click.

Computer Requirements

The software runs on Intel PC's equipped with Windows 95, 98 or NT. It is self-contained, requiring no additional programs or tools to run.

The simulations run much faster than you might expect for such detailed models. On any current PC, the programs run faster than real time. On higher-end desktop computers, the vehicle models run several times faster than real time. In other words, a run simulating a 10-second test will finish in just a few seconds.

Working with SIMULINK® and MATLAB®

SIMULINK® is a software package for modeling, simulating, and analyzing dynamical systems in general. It runs under MATLAB®, a mathematical workshop. SIMULINK and MATLAB are available from the MathWorks, Inc. SIMULINK provides a graphical user interface for building models as block diagrams. The graphical interface is popular for

developing dynamical models for many fields, such as electronics, hydraulics, chemistry, etc.

SIMULINK is not particularly useful for building equation sets for complex mechanical 3D systems such as the CarSimEd models. However, it includes S-functions (system functions) to augment and extend the building blocks in SIMULINK to include arbitrary complex systems. The S-function appears in a SIMULINK model as a block in the block diagram.

The mathematical behavior of S-functions can be defined either as a MATLAB M-file, or as an executable piece of object code in the form of a DLL (dynamic link library) obtained by compiling C or FORTRAN source code. Such "executable" functions are called MEX files (where the EX stands for executable). The S-functions in CarSimEd are made from C source code and are thus called CMEX functions.

CarSimEd includes DLL files that can be loaded and run by SIMULINK. (The DLL files were created by compiling CMEX files with the same equations of motion as used for the stand-alone C programs used to create the EXE solver programs in CarSimEd.) The simulations can be run from within SIMULINK, using the SIMULINK integrators and the SIMULINK environment for setting control inputs to the vehicle model. You can build controller models (steering, braking, etc.) in SIMULINK and test them with the full nonlinear CarSimEd vehicle models.

Note: SIMULINK and MATLAB are not included with CarSimEd—they must be obtained from the MathWorks. If you are not already using SIMULINK, don't worry. CarSimEd is fully functional as a stand-alone package. SIMULINK is not needed to make runs or view results.

The extra capability added by SIMULINK is mainly the ability to combine new controller designs with the CarSimEd models.

How CarSimEd Works

CarSimEd combines information from the data screens with vehicle dynamics programs to simulate the vehicle behavior. CarSimEd also links the simulation results with animation and plotting programs. The CarSimEd package is primarily made up of four tightly integrated software modules, as shown schematically in Figure 1.1 and described below.

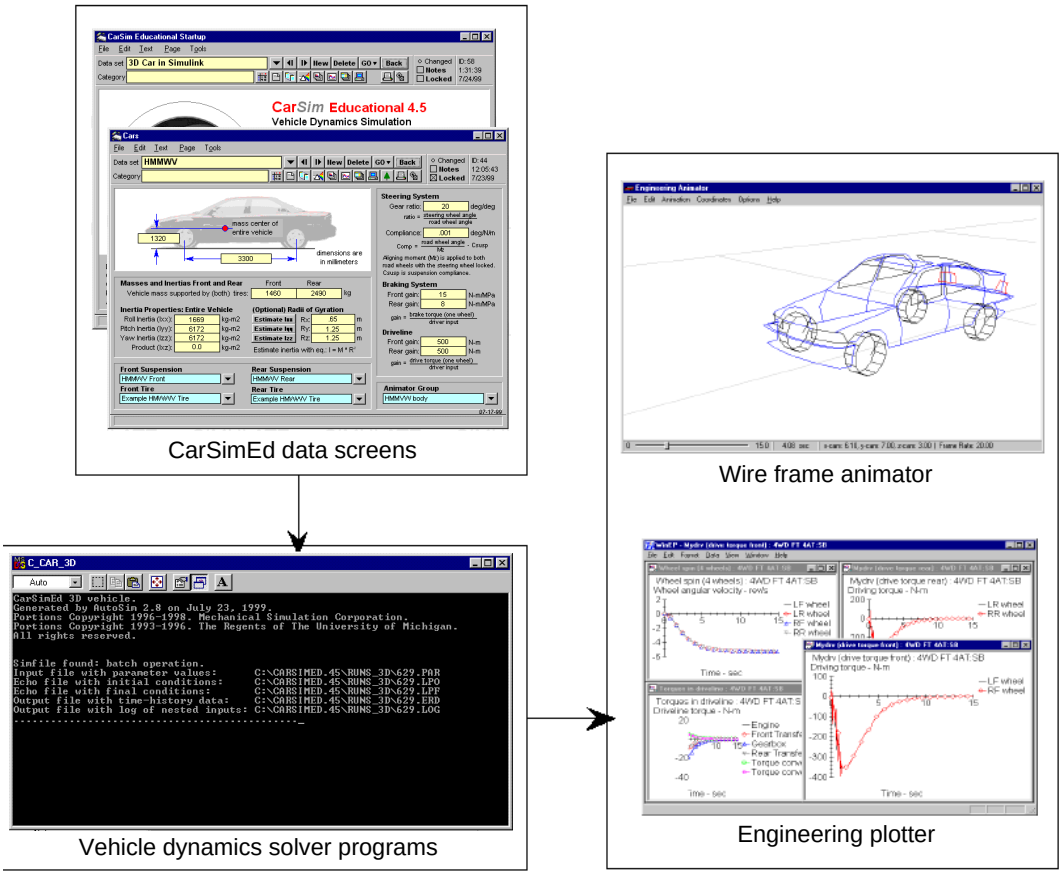


Figure 1.1. Four parts of CarSimEd.

1. Data screens serve as your primary interface to CarSimEd. They contain vehicle model parameters, control inputs, and run settings. The data screens are part of a database that maintains libraries of related data sets. CarSimEd includes about 30 libraries (each with multiple data sets) that are linked together to make up the database.

2. Vehicle dynamics solver programs use equations of motion in mathematical models to calculate output variables. The process of performing these calculations is called making a "simulation run" or simply a "run." CarSimEd includes the models in two forms: (a) as stand-alone EXE application files, and (b) as DLL plug-in files (also called CMEX files) for use with SIMULINK.

3. A wire-frame animator shows the resultant vehicle motions. You can view the simulated motions, zoom in and out with a simulated camera, and interactively move around the simulated vehicle to change your point of view.

4. The Windows Engineering Plotter (WinEP) creates plots of vehicle variables as functions of time or as cross plots of output variables. Use this tool to view any of the hundreds of variables computed by the simulation models. Plot any combination of variables and overlay plots from different runs. If test measurements are available, you can overlay simulation and test results.

What CarSimEd Does Not Do

CarSimEd is not a general-purpose simulation tool. It is customized for the kinds of simulations described above. Simulations with other models are not possible without adding programs. Some limitations of the models in CarSimEd are:

- The 3D vehicle model does not support closed-loop speed control for variable speed.
- The suspension springs, dampers, compliance, and kinematics are modeled with linear coefficients. There are no bump stops.
- The handling model assumes constant steer ratios (no Ackerman effect.)
- The brakes are modeled with linear gains between brake torque and brake input.
- The tire model does not allow specification of forces and moments for large slip angles. Large slip is handled using built-in functions that are reasonable, but which cannot be adjusted by the user.
- The road surface is smooth and level with a constant friction coefficient.
- CarSimEd has no driveline dynamics. Throttle inputs are related to drive torque by constant coefficients. The ride model assumes a constant vehicle speed.
- The vehicle models do not have aerodynamic effects.
- The five-link suspension model does not include compliance.

A commercial version of the software is available with more extensive modeling, including a detailed driveline model. The braking and handling model has nonlinear steering geometry, nonlinear brake torque vs. pressure, nonlinear suspension geometry (toe and camber), nonlinear springs, nonlinear dampers, and nonlinear differentials. A more detailed tire model is built-in that can match experimental data over a wide range of conditions. Friction and ground elevation are specified as functions of ground position. Aerodynamic effects are included. Closed-loop controllers are available for variable speed and steering to follow a prescribed path and speed.

To obtain information about the commercial version of CarSim, please contact MSC (or view the web site):

Mechanical Simulation Corporation
709 West Huron Street, Suite 50
Ann Arbor, MI 48103
tel. (734) 668-2930 fax (734) 668-2877
email: info@trucksim.com
web: http://www.trucksim.com

Notational Conventions in This Manual

Some standard conventions are used throughout this manual to make it more concise, while still being reasonably easy to read.

Fonts and Type Face

The bulk of this manual is presented using a Times font. Other fonts and styles are used to convey special meanings.

- The `Courier` font is used for keywords (keywords are special names used in CarSimEd files) and names of computer files and folders. It is also used to designate the contents of text files if they're not too long. (Long text files are shown in the regular Times font because it is more compact and easy to read.)
- **Bold** is usually used for things that you, the user, might click on: buttons, menus, menu items, etc. It is used for text that is shown in bold on the screen, such as titles of data sets. It is also used for section names (e.g., "Section **Animator: Shapes**").
- Underline is used to indicate text that you, the user, might type. For example, "In the speed field, type 100."
- *Italics* are used to indicate variable names and place-holders. E.g., *id*.LPO represents a file name where *id* is some value. If *id* were 171, then the file name would be `171.LPO`.

Pathnames

Pathnames are specified relative to the `CarSimEd` folder. For example, suppose the root folder is `C:\CarSimEd`. The folder `C:\Runs\Runs.tbk` would be named as `Runs\Runs.tbk`.

Terminology

Specialized terms are defined in the first two appendices. Appendix A is a glossary of words used throughout this manual in a manner that is specific to CarSimEd (e.g., keyword, run, library). The glossary also defines file types and names of software packages that might otherwise be unfamiliar (e.g., CMEX, ToolBook, SIMULINK).

Appendix B defines specialized terms applicable to vehicle dynamics. It also defines the coordinate systems, axes, and sign conventions used in the CarSimEd models.

2. CarSimEd Installation

This chapter describes how CarSimEd is installed on your computer. It also describes how CarSimEd works with SIMULINK. (SIMULINK is a separate software package from the MathWorks that is obtained and installed independently of CarSimEd.)

Computer Requirements

Operating Systems

The complete software package runs under Windows 95, 98, and NT.

> Note: The vehicle dynamics solver programs in CarSimEd are not specific to Windows. Similar versions have been compiled and run on other operating systems such as UNIX, Windows 3.1, DOS, and MacOS. However, the animator, plotter, and graphical interface are currently available only for Windows 95, 98, and NT.

Networks

In general, CarSimEd should not be run from a remote server. If CarSimEd is posted on a network server, you should copy the entire directory to your disk and run the local copy. Installation is simple: just copy the root CarSimEd directory and follow the instructions in Chapter 10 in the section **Installing CarSimEd in a New Directory**.

Although CarSimEd should not be run completely from a remote server, it is possible to access data from a server. For example, you might have several standard vehicle data sets on the server that can be used as inputs for runs made on local machines. However, there are a few restrictions.

1. The network **must** be configured such that the drive containing CarSimEd appears with the same volume name for all users (e.g., drive `R:`). (Portions of CarSimEd use full pathnames, so it is essential that each pathname be valid for all users.)
2. It is recommended that all library files on the server be locked, and that all users make runs from a local copy of CarSimEd, including their own **Runs** library.
3. Whenever a screen appears showing data from the CarSimEd database, the corresponding file is inaccessible to all other potential users. Therefore, only one person at a time can access data from the server.

Installation of CarSimEd

The CarSimEd software is provided as a self-extracting EXE file that is typically downloaded from the MSC web site. Run the EXE file to unpack a folder `INSTALL`, that contains a file `SETUP.EXE`. Run `SETUP` to install the software. When asked, select the full installation. When asked about installing program manager groups, click the **YES** button. The program then runs through an initialization process that takes a minute or two.

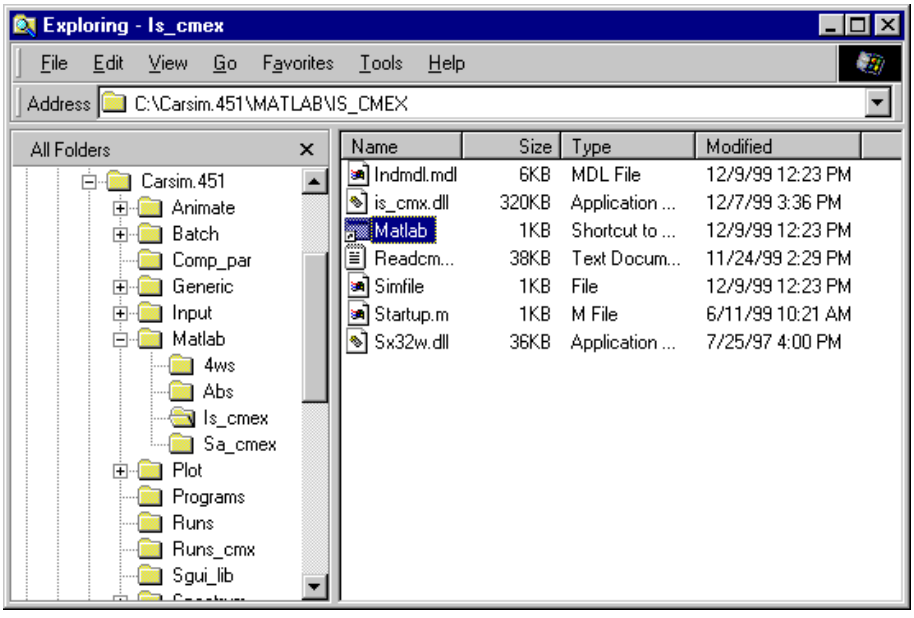
If you move the main folder (e.g., `C:\CarSimEd.45`), it is necessary to re-initialize the software as described in chapter 10.

Linking to SIMULINK and MATLAB

SIMULINK and MATLAB must be installed on your computer if you wish to combine the CarSimEd and SIMULINK packages.

Updating the MATLAB PIF

The integration between CarSimEd and SIMULINK is done with software contained in the folder `Matlab`, which in turn contains one folder for each SIMULINK model. Each SIMULINK model folder contains a shortcut to the MATLAB program, a CarSimEd DLL solver module, and some extra support files.



Each of these folders contains one file that you might have to modify: `Matlab.pif`. The Program Information File (PIF) acts as a pointer (shortcut) to the MATLAB program. When MATLAB is started by opening the PIF, the default directory is the directory containing the PIF, rather than the normal MATLAB default. Depending on

your Windows settings, the PIF might be displayed without the PIF extension (see the next figure).

> Note: The PIF was introduced in Windows 3.1 to support DOS programs running in the Windows environment. It is supported in Windows 3.1, 9x, and NT, and is also supported by the ToolBook software used to manage the CarSimEd database.

When the MATLAB folder in CarSimEd is installed on your computer, the `Matlab` PIF in each model folder points to the default location shown in Figure 2.2. You will have to change the Command Line field if your MATLAB installation is not standard. To do so:

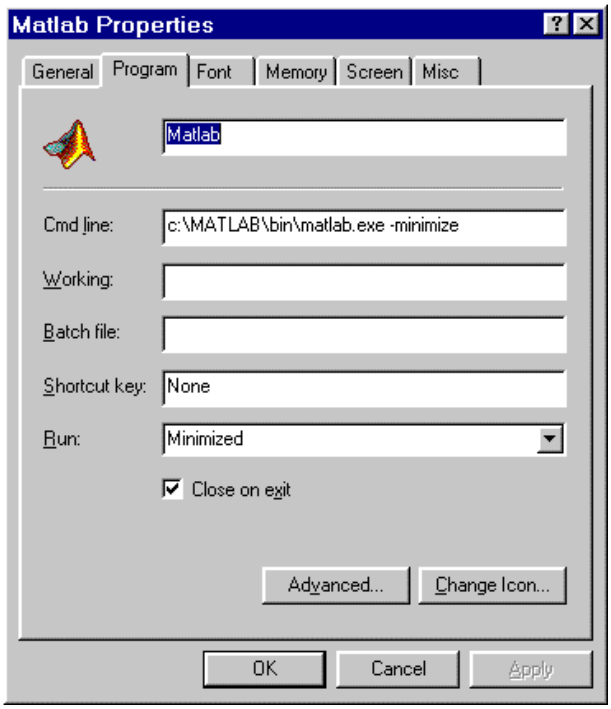


Figure 2.2. Properties of the Matlab PIF.

1. In the Windows explorer, locate the MATLAB PIF file.
2. Right-click on the `Matlab.pif` (depending on your settings, the file might appear simply as `Matlab`). From the pop-up menu, select the item **Properties**. A properties window will appear (see Figure 2.2).
3. In the Properties window, select the **Program** tab (see Figure 2.2).
4. Change the pathname in the Command Line field to the location of MATLAB on your hard drive.
5. Click the OK button to close the window.
6. Repeat steps 1 through 5 for all other MATLAB PIFs.

MATLAB Versions

CarSimEd 4.5 has been tested with SIMULINK/MATLAB versions 5.2 and 5.3.

The first time you make a run in SIMULINK, you might get a stream of warnings in the MATLAB window. This is because the SIMULINK MDL file was created with a different version of SIMULINK than the one you are using (version 5.2 vs. 5.3). To prevent this in future runs, save the MDL file. In doing so, SIMULINK makes the file fully compatible with your version.

3. Database Organization

You will generally spend most of your time interacting with the CarSimEd database. This chapter provides an overview of how the database is organized.

Introduction to the CarSimEd Database

Throughout this manual, the database is considered at five levels. Starting with what you see on the screen, the levels are:

1. **Data screens**. Any screen display in CarSimEd that has editable fields or other forms of user settings is called a data screen.
2. **Data sets**. A data set is the information you provide and can edit in a data screen, minus the screen itself. The data screen is a view of a data set as seen through the CarSimEd user interface.

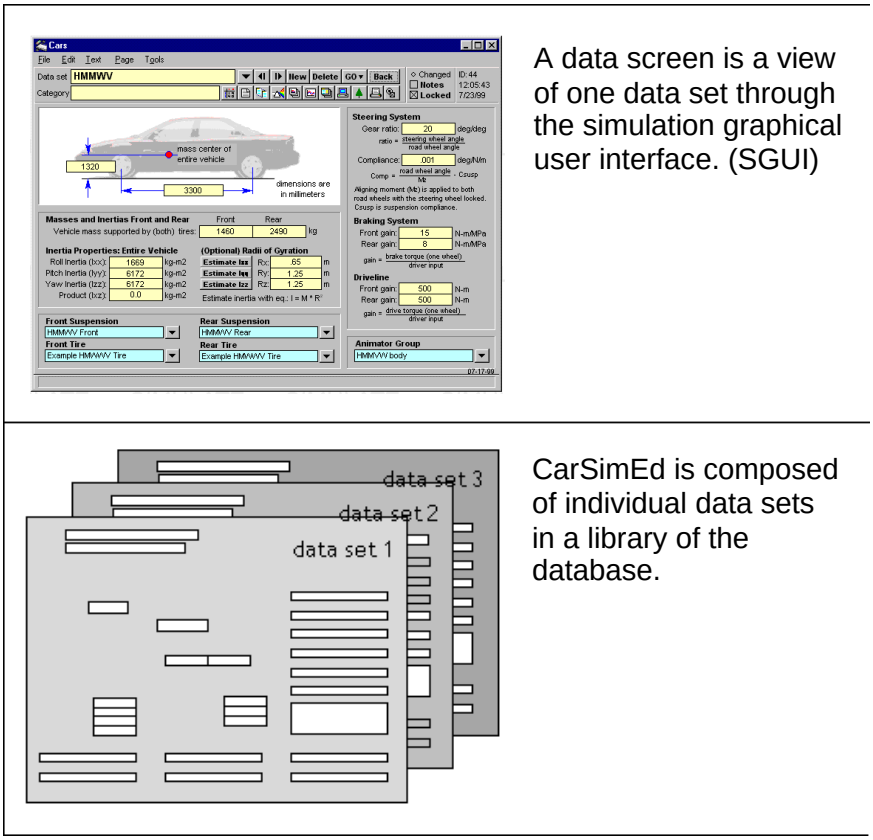

Cars
File Edit Text Page Tools
Data set HMMWV
Category
New Delete GO Back
Changed ID:44
Notes 12:05:43
Locked 7/23/99
mass center of entire vehicle
1320
3300
dimensions are in millimeters
Steering System
Gear ratio: 20 deg/deg
Compliance: .001 deg/Nm
Aligning moment (Mz) is applied to both road wheels with the steering wheel locked.
Ccusp is suspension compliance.
Masses and Inertias Front and Rear
Front Rear
Vehicle mass supported by (both) tires: 1460 2490 kg
Inertia Properties: Entire Vehicle
(Optional) Radii of Gyration
Roll Inertia (Ixx): 1669 kg-m2
Pitch Inertia (Iyy): 6172 kg-m2
Yaw Inertia (Izz): 6172 kg-m2
Product (Ixz): 0.0 kg-m2
Estimate Ixx Rx: .65 m
Estimate Iyy Ry: 1.25 m
Estimate Izz Rz: 1.25 m
Estimate inertia with eq.: I = M * R²
Braking System
Front gain: 15 N-m/MPa
Rear gain: 8 N-m/MPa
Driveline
Front gain: 500 N-m
Rear gain: 500 N-m
Front Suspension
HMMWV Front
Rear Suspension
HMMWV Rear
Front Tire
Example HMMWV Tire
Rear Tire
Example HMMWV Tire
Animator Group
HMMWV body
07-17-99
A data screen is a view of one data set through the simulation graphical user interface. (SGUI)
data set 3
data set 2
data set 1
CarSimEd is composed of individual data sets in a library of the database.

Figure 3.1. Contents of a CarSimEd Library.

3. **Libraries**. A library is a collection of one or more data sets, plus the information needed to provide a view in the user interface. Figure 3.1 shows the contents of a library: multiple data sets, plus a graphical view of one data set at a time.
4. **The CarSimEd database**. CarSimEd includes about 30 libraries.
5. **The Simulation Graphical User Interface (SGUI)**. The architecture of CarSimEd is called the SGUI. It includes the database, the code for managing the database, plus utilities such as the plotter and animator.

CarSimEd automatically handles the transfer of information between the libraries and solver programs. You do not have to know where the libraries are (file names) or what they contain (file formats). However, it is important to understand the distinction between data screens, data sets, and libraries. Specifically, you should understand that:

- Each CarSimEd data screen shows one data set from the library associated with the screen title.
- Changes made on a data screen affect the one data set, but all other data sets in the library are unaffected.
- Links between data sets can be made and broken as you see fit. In this way, vehicles, components, inputs, etc. can be rapidly swapped.
- When a data set is changed, simulation results will be affected only if:
  a. the changed data set is linked to the run, and
  b. the simulation is run (again) after the changes were made.

Data Screens

Common elements in a data screen are introduced for the example screen shown in Figure 3.2.

Each data screen has three kinds of elements that you use:

1. **Yellow fields**. These contain data that you can edit directly. For example, to change the vehicle wheelbase, find the yellow field (3) with the current wheelbase value, click on the field, and change the value using the mouse and keyboard.
2. **Buttons.** All of the library screens include buttons at the top to quickly navigate through the data sets in the library, and to go to other libraries and programs in CarSimEd (see the buttons near (2)). In addition, some of the libraries have special buttons for functions such as running a simulation, running an animation, etc.
3. **Blue links** with pull-down menus**.** These buttons are always marked with a triangle (e.g., item (4)). This indicates that if you click over the button with the left mouse button then a pull-down menu will appear.

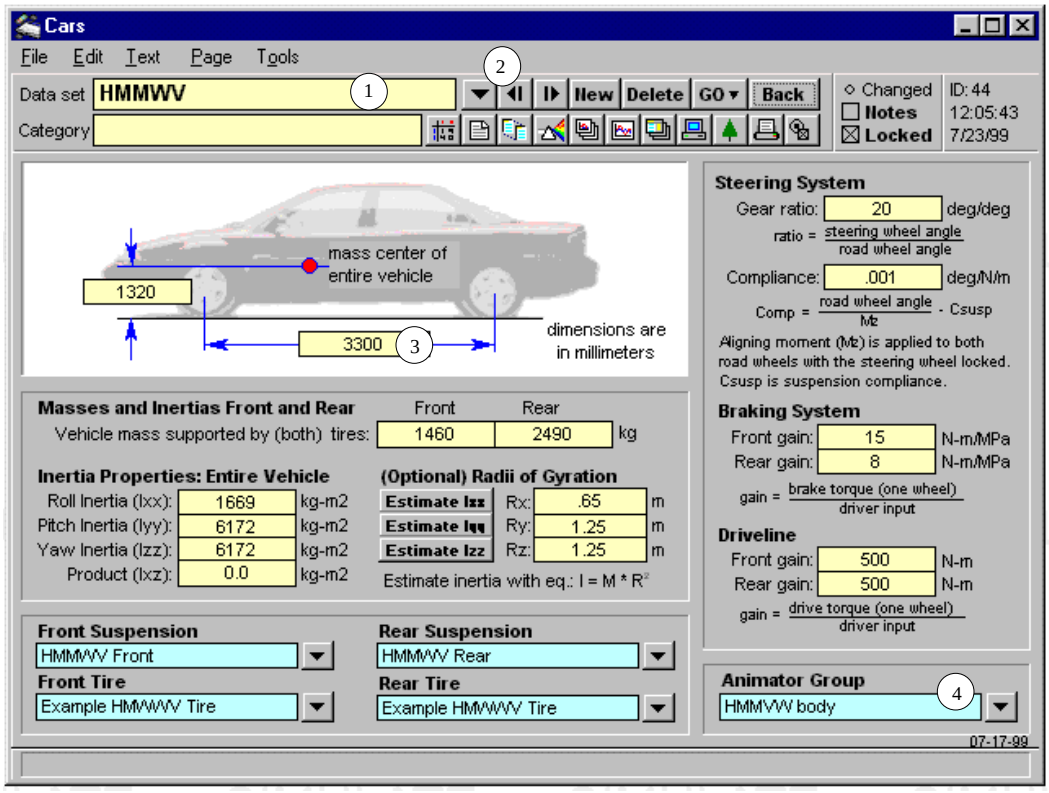
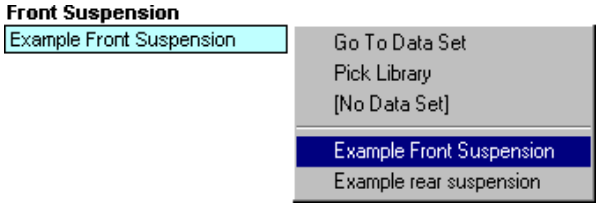


Figure 3.2. Example data screen.

The yellow field in the upper-left corner of the screen is the title of the data set, e.g., **HMMWV** (1). The title is just text that you choose to identify this data set in the library. It has no significance to the solver program. (However, it can have great use for organizing your data sets.)

Each blue field represents a link to another data set. The name of the data set is shown in the blue field, and the triangle button is used to display a pull-down menu. For example, click the triangle button (4) to view the following menu.



The menu can be used to select a different data set in the linked library (choose the data set of interest from the list displayed at the bottom of the menu). It can also be used to go to the linked data set (choose the menu option to **Go To Data Set**).

After using **Go To Data Set** to go to a different library, you can return to the previous screen using the **Back** button, shown below, located in the upper-right corner of each data screen.

The standard elements in a data screen, such as the ribbon bar at the top, and the blue links, are described in more detail in Chapter 8.

Making New Data Sets

Every CarSimEd screen includes a **New** button in the ribbon bar. This is one of the most essential buttons in the SGUI — click it to make a copy of the current data set.

One of the reasons that CarSimEd can be such a productive tool is that data sets do not have to be re-entered. Rather than modifying an existing data set, the normal practice is to make a new copy and modify the copy. This way you can switch back to the original whenever necessary.

On-Line Help

Every CarSimEd screen includes a status bar at the bottom of the screen. Move the cursor over an object to read a one-line description of its function in the status bar.

A Map of the CarSimEd Libraries and Screens

Except when using CarSimEd with SIMULINK, all information used by the CarSimEd solver programs is obtained from the data sets seen in the CarSimEd data screens. The plotter and animator also work from the same data.

Each screen design is associated with a separate library file. CarSimEd includes the following types of libraries:

- vehicle parameters,
- control and environmental inputs for simulation runs,
- run setups including simulation and post processing control,
- plot setups,
- animation setups, and
- batch controls.

The top level of CarSimEd is shown graphically in Figure 3.3

You normally start CarSimEd at the **Startup** screen. Clicking the **Start** button on that screen takes you to the **Runs** screen. The **Runs** screen is the window to a library of many run descriptions, each with links to vehicles, inputs, and various settings.

The **Runs** screen also has three buttons for:

- running new simulations,
- using the plotter to view simulation results, and
- using the animator to view simulated motions.

Figure 3.4 shows a partial map of the various libraries (screens). For example, from the **Startup** screen, you can go "down" to a **Runs** screen. From there, you select the various inputs (braking, steering, etc.), the vehicle, and the parameters that control the numerical solution methods used to make a simulation. After the simulation is run, you specify the plot and animation setup to view results.

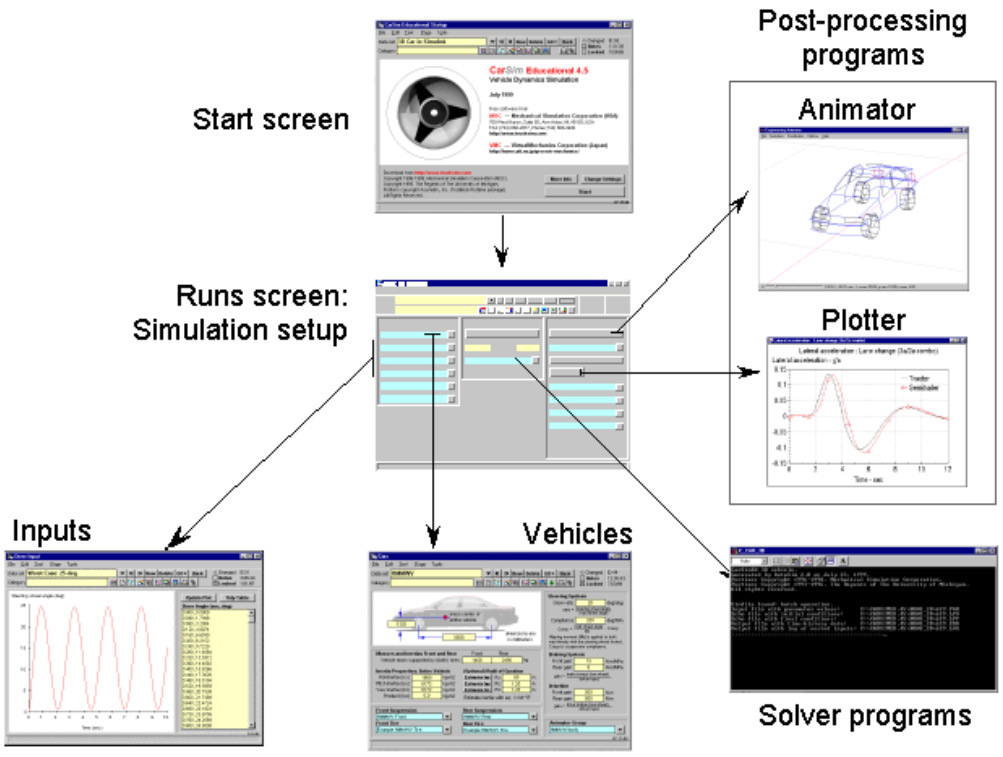
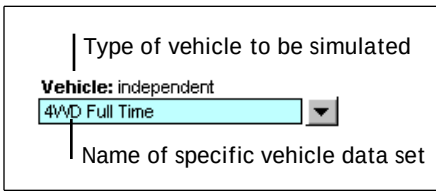


Figure 3.3. Role of Runs screen in CarSimEd.

In the map, data sets from different libraries are connected with links. For example, here is a link between the **Runs** library and a vehicle library:



In general, libraries shown near the top of the map are more systems-oriented, and libraries shown near the bottom are more component-oriented.

Map for the 3D Handling Model

Figure 3.4 shows a more detailed map of the various libraries (screens) that pertain to the 3D handling model. For example, from the **Startup** screen, you can go "down" to the **Runs** screen for the 3D handling model. From there, you select the various inputs (brake, steer, etc.), the vehicle, and the parameters that control the numerical solution methods used to make a simulation. After the simulation is run, you specify the plot and animation setup to view results.

> Notes: In order to create the detailed map at a reasonable size, the names were shortened from the full screen titles.
>
> Libraries used for more than one of the simulations are shown in italics.

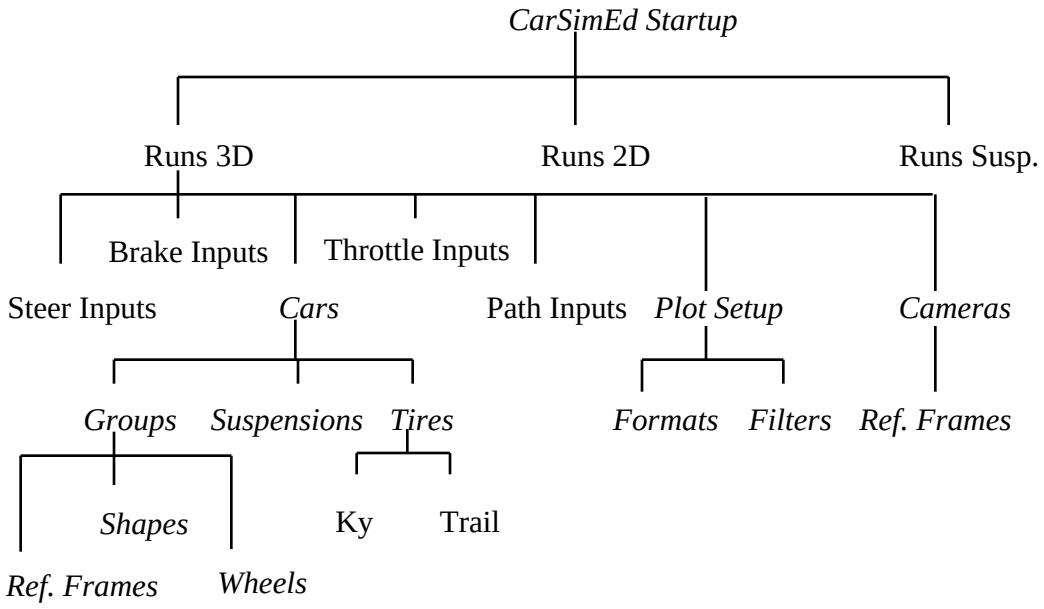


Figure 3.4. Partial map of the CarSimEd libraries.

Map for the 2D Ride Model

Figure 3.5 shows a map of the libraries for the 2D ride model. Notice that nearly all of the library names are in italics, indicating that they are also used for 3D handling/braking simulations. The main difference is with the inputs: the steer, braking, and throttle inputs for the 3D model are replaced with a road profile input. Also, some of the tire data required for the 3D model are not required for the 2D simulations. (The extra data are ignored when the simulation runs.)

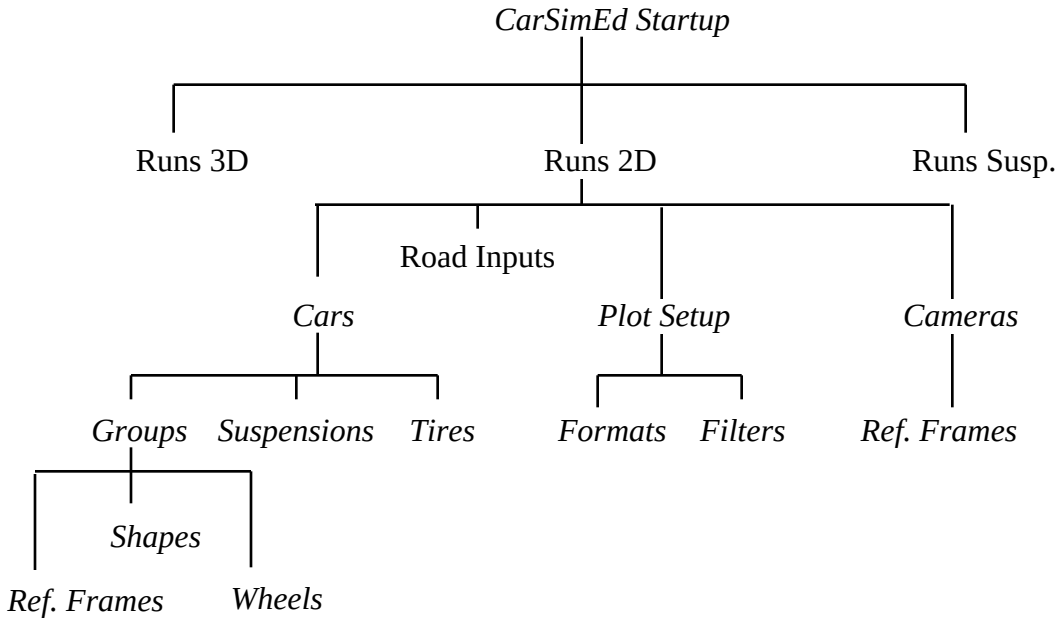


Figure 3.5. Partial map of the CarSimEd libraries.

Map for the 3D Suspension Model

Figure 3.6 shows a map of the libraries for the 3D suspension model. This model does not use any of the vehicle data, but shares the libraries for the animator and plotter.

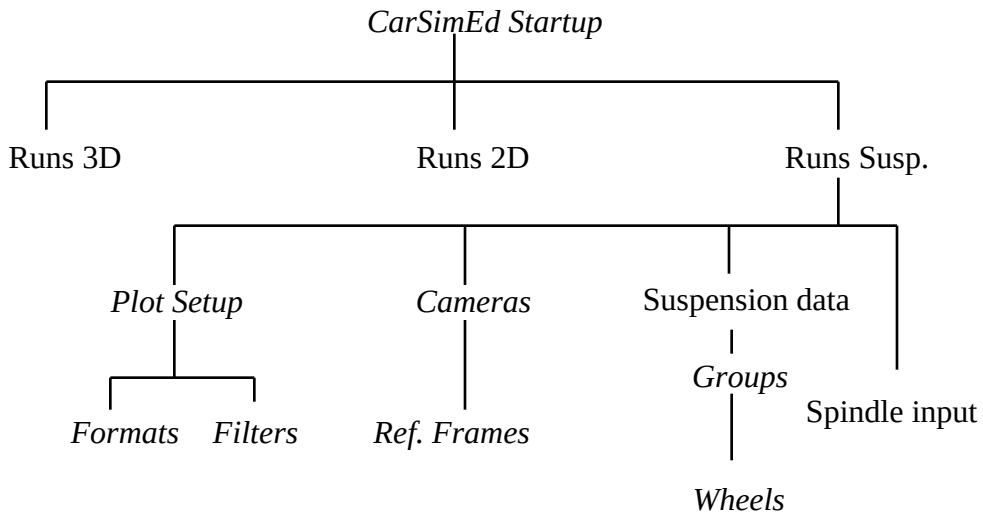


Figure 3.6. Partial map of the CarSimEd libraries.

How CarSimEd Communicates with SIMULINK

The SIMULINK version of a CarSimEd solver program works almost the same as the stand-alone version. It has the same equations of motion, the same model parameters, and the same control inputs and disturbances.

All of the files that are read and written by a standard CarSimEd solver program are also read and written by the SIMULINK version. Therefore, the CarSimEd post-processing plotter and animator work nearly the same as with the stand-alone version.

> Note: There is just one mathematical difference in the way the SIMULINK program (a DLL file compiled from CMEX source code) runs relative to the stand-alone program (an EXE file compiled from C source code).
>
> In SIMULINK, the numerical integration is controlled by the MATLAB/SIMULINK routines.
>
> The CarSimEd DLL will generate an error message if you try to use a variable-step integrator. A fixed-step integrator is required, and the ODE2 (Heun) is recommended because it runs the fastest for a given level of accuracy.

In addition to the normal inputs from the SGUI and the output files for the plotter and animator, the CMEX program exchanges information with SIMULINK. Figure 3.7 shows the the flow of information. The CMEX program receives some inputs from SIMULINK and provides output variables to the MATLAB/SIMULINK workspace. The outputs are identical to the variables that are written into the output ERD file for plotting and animation.

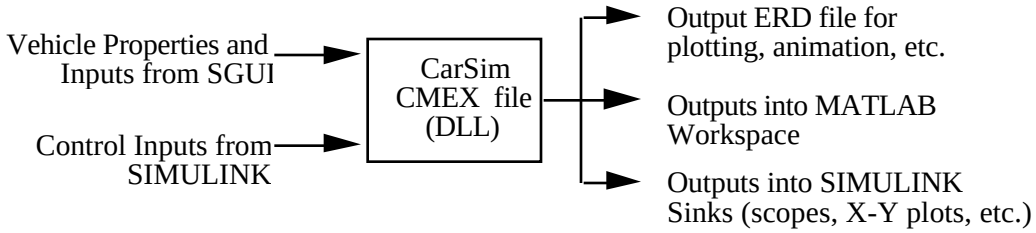


Figure 3.7. CMEX inputs and outputs.

The communication between the CMEX S-function and SIMULINK follows the S-function convention and is made at each time step through predetermined inputs and outputs.

Communication between the CMEX program and the CarSimEd database is made in the same fashion as for the stand-alone EXE program (see Chapter 5).

A README text file is included in each MATLAB folder to describe the specific input and output variables for the model associated with the folder. Portions of a README file are presented in Appendix J.

The CarSimEd database includes four **Runs** libraries. Three are for stand-alone simulations (without SIMULINK) and one is for SIMULINK. The screens are nearly identical in appearance: the main differences are (1) the screens have different window titles and (2) the SIMULINK-compatible screen has two buttons for accessing SIMULINK, whereas the stand-alone screens have just one button for running the EXE solver program.

4. The Basics of Using CarSimEd

This chapter explains the basics of how to use CarSimEd. It assumes no prior knowledge other than what has been covered in the preceding chapters. Each section covers a specific task, and includes step-by-step instructions. The sections are fairly compact, covering just the essence of how to accomplish each task. (The detailed reference information is covered in following chapters.)

After the reference material in Chapters 5 through 9, Chapter 10 continues by explaining how to do more advanced tasks in CarSimEd.

About the Runs Screen

Figure 4.1 shows the **Runs** screen — the screen most central to the operation of CarSimEd.

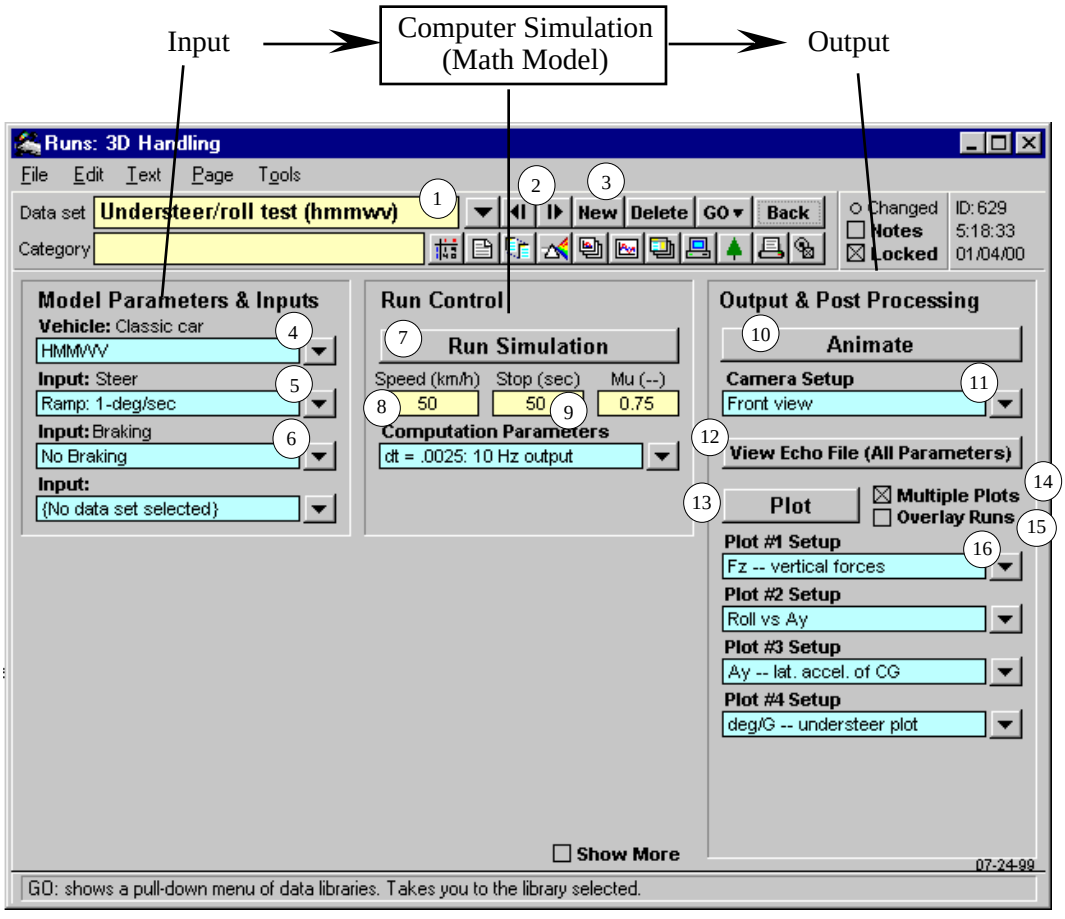


Figure 4.1. The Runs screen.

The settings in the left-hand portion of the screen give you access to the inputs for the mathematical vehicle models used to simulated tests; the settings and buttons in the right-hand region give you access to views of the simulation outputs; and the settings and button in the center are used to run the simulated test.

The circled numbers will be used in the following sections to describe common actions. (A different Runs screen is used for making runs with SIMULINK, as shown in Figure 4.3 on page 35.)

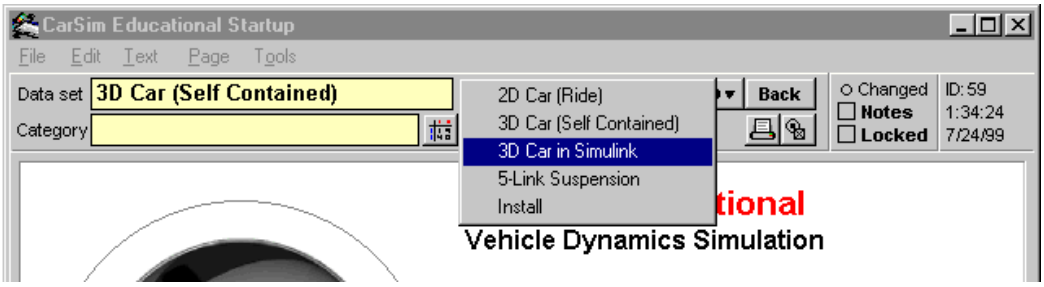
> Notes: Figure 4.1 shows one possible appearance of the **Runs** screen. It is obtained when the **Overlay Runs** and **Show More** boxes are not checked.
>
> CarSimEd includes four **Runs** screens: three for stand-alone solver programs, and one for working with SIMULINK. The screens are nearly identical (compare Figure 4.1 and Figure 4.3). There are many references throughout this manual to a generic **Runs** screen. Unless otherwise specified, these references apply to all versions.

Getting to a Runs Screen

Most operations begin with one of the four **Runs** screens in CarSim Ed.

When Starting

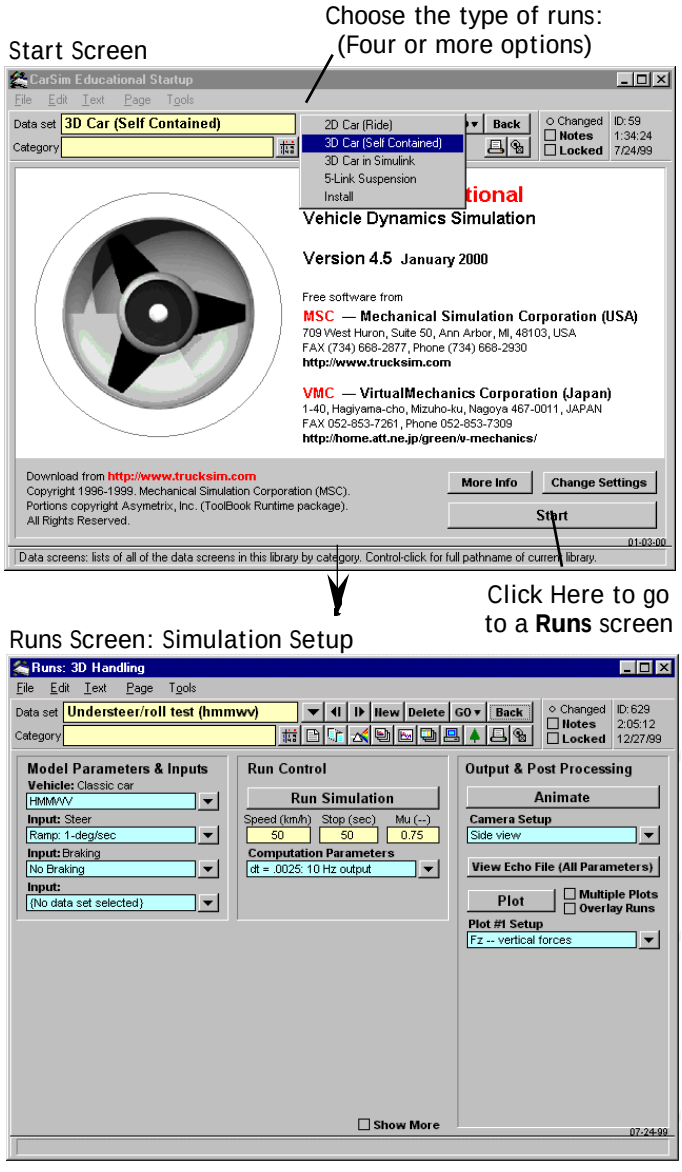
1. Start CarSimEd to get to the **Startup** screen.
   a. Use the Windows Start menu, or
   b. open the file `Startup.tbk` from within the `CarSimEd` folder (or a shortcut to that file)
2. Use the triangle button next to the data set title in the ribbon bar to go to the start point for the type of simulation with which you are interested (see the figure below).



Choose either:

- **2D Car (Ride)**, OR
- **3D Car (Self Contained)**, OR
- **3D Car in Simulink**, OR
- **5-Link Suspension**.

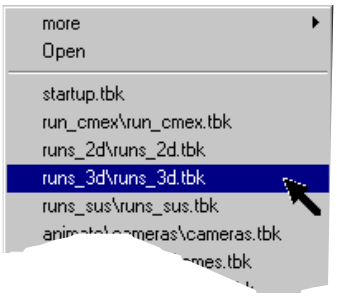
3. Click the **Start** button in the lower-right corner of the screen.



After Starting

All CarSimEd screens include a standard ribbon bar with navigation buttons. The GO button displays a pull-down menu when clicked.

1. Click the **GO** button in the ribbon bar (GO▾) to display a pull-down menu.



2. Select the desired **Runs** screen from the menu.

From the Plotter or Animator

The CarSimEd plotter and animator are independent programs. When you launch one of them by clicking a button on the **Runs** screen, the **Runs** screen is still there. However, it might be covered by the windows of the other programs, especially if you are not using a large computer monitor.

The recommended way to return to the **Runs** screen from the plotter or animator is to exit the program.

1. To exit the plotter or animator, click the ✕ box in the upper-right corner of the WinEP or Animator screen.

Making a New Run (Stand-Alone)

Use this method to make a new run in CarSimEd using a stand-alone solver program (without SIMULINK). Existing runs are left intact and their output can still be viewed with the plotter and animator.

Start from the **Runs** screen (see Figure 4.1 on page 30).

1. Using the navigation buttons (2), find an existing data set that is close to what you want. The main thing to look for is whether there is a steering input (steer vs. time) or a path input (e.g., follow a circle), because the two are mutually exclusive.
2. Click the **New** button (3) in the ribbon bar.
3. Inspect and edit the simulation inputs.
    a. Select the vehicle of interest from the linked vehicle library (4).
    b. Select a steering or path input from the linked library (5).
    c. Select a braking and/or throttle input from the linked library (6).
    d. Specify the speed using the yellow field (8).

e. Specify the simulation stop time (9).

4. Click the **Run Simulation** button (7).

A console application window (DOS-style) appears while the simulation runs. A bar graph at the bottom of the window indicates the progress. When the simulation finishes the DOS window disappears and you are returned to the **Runs** screen.

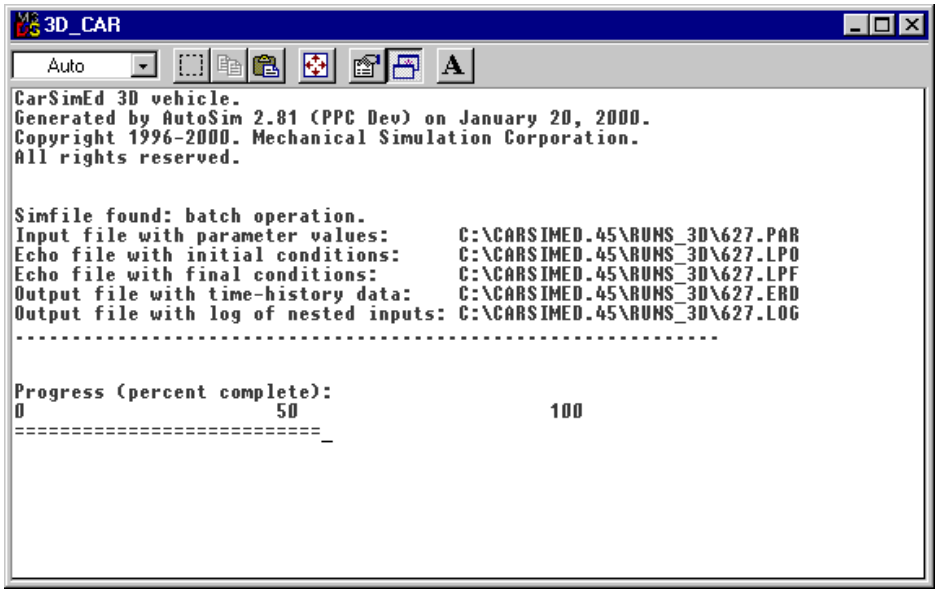

3D_CAR
Auto
CarSimEd 3D vehicle.
Generated by AutoSim 2.81 (PPC Dev) on January 20, 2000.
Copyright 1996-2000. Mechanical Simulation Corporation.
All rights reserved.
Simfile found: batch operation.
Input file with parameter values: C:\CARSIMED.45\RUNS_3D\627.PAR
Echo file with initial conditions: C:\CARSIMED.45\RUNS_3D\627.LPO
Echo file with final conditions: C:\CARSIMED.45\RUNS_3D\627.LPF
Output file with time-history data: C:\CARSIMED.45\RUNS_3D\627.ERD
Output file with log of nested inputs: C:\CARSIMED.45\RUNS_3D\627.LOG
Progress (percent complete):
0 50 100

Figure 4.2. Console display when solver program is running.

Notes: If the vehicle comes to a complete stop, or rolls over, the normal settings cause the program to quit after updating all output files.

There are a few other occasions when the CarSimEd solver program will not run, or will quit prematurely. Among them are:

1. The output file cannot be written if it is in use by the plotter or the animator (this can happen if you are replacing a run, as described in the next section).
2. In rare circumstances, a bad input parameter will cause the program to quit before it can create any output files. However, in most cases, the program will generate the output files even if it quits suddenly.

Replacing a Run (Stand-Alone)

Use this method to replace an existing run. Start from the **Runs** screen (see Figure 4.1 on page 30).

1. Find the existing data set in the **Runs** library that you want to replace. Use the navigation buttons (2) next to the Data Set (title) box (1).
2. Inspect and edit the simulation inputs (vehicle, steering or path input, braking or throtthle input, speed, etc.)
3. Click the **Run Simulation** button (7).This makes a run, as described in the previous section, except that in this case the previous run (selected in step 1) is over-written.

Making the First SIMULINK Run

Use this method to start up a SIMULINK model that uses a CarSimEd CMEX module. If SIMULINK is already running, see the next section.

Start from the **Runs SIMULINK** screen (see Figure 4.3).

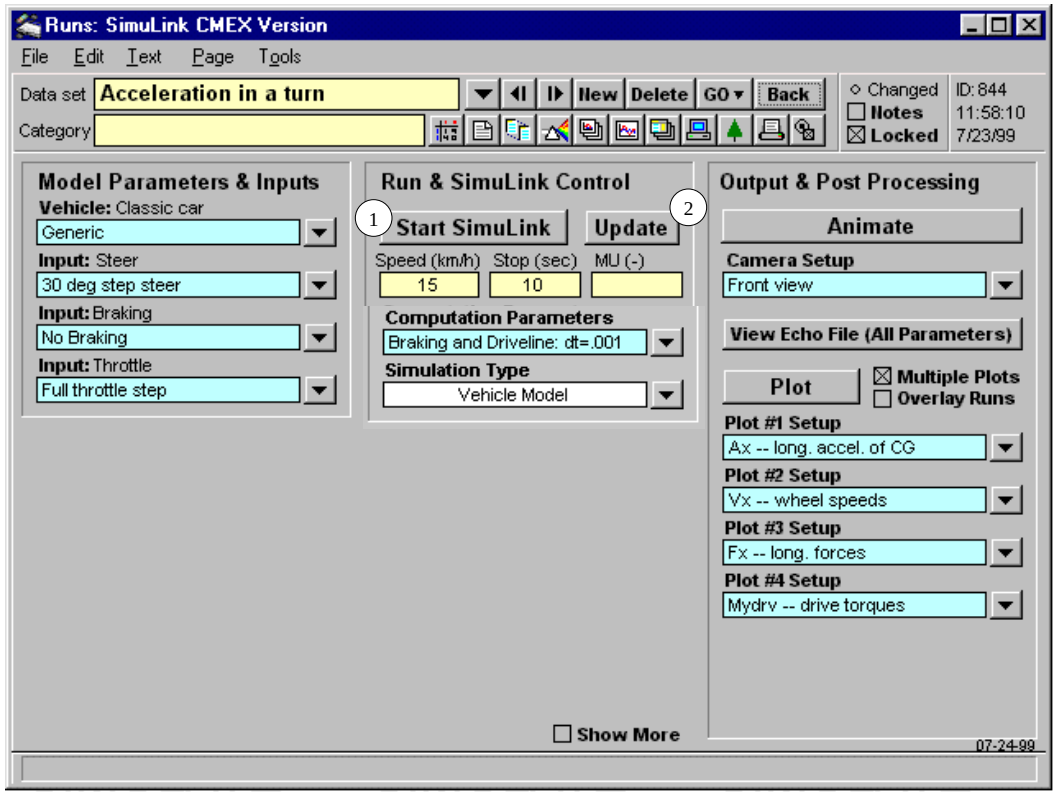


Figure 4.3. The CMEX Runs screen.

Notes: The figure shows one possible appearance of the **Runs** screen. It is obtained when the **Overlay Runs** and **Show More** boxes are not checked.

If you compare Figure 4.1 and Figure 4.3, you will see that the only differences are in the numbered items (1) and (2) (in Figure 4.3).

1. Prepare to make a run, as described earlier for the stand-alone version (page 30).
2. Click the **Start SIMULINK** button (1).

   You should see the splash screen for MATLAB, and then a window with a SIMULINK model, similar to the display shown in Figure 5.2 on page 57.

   You are now through with the CarSimEd part of the process.
3. Click in the SIMULINK window to make it active.
4. Type Ctrl+T to start the run. Or, select the **Start** menu item from the **Simulate** menu.
5. When the run is complete, you can view the results with SIMULINK and MATLAB tools or with CarSimEd tools. (The CarSimEd tools are described in the rest of this chapter.)

Making Additional SIMULINK Runs

Use this method to make a run when SIMULINK and MATLAB are already running in the background.

Start from the **Runs SIMULINK** screen (see Figure 4.3).

1. Prepare to make a run, as described earlier (page 30).
2. Click the **Update Data** button (2). This copies the specifications for the run into a batch control file that will be used by the S-function in the SIMULINK model.

   You are now through with the CarSimEd part of the process.
3. Click in the SIMULINK window to make it active.
4. Type Ctrl+T to start the run. Or, go to the **Simulate** menu and select the **Start** item.

Viewing an Animation

Start from a **Runs** screen (see Figure 4.1 on page 30). (The simulation run must have already been made in order to view an animation.)

1. Choose a camera setup using the blue link (11). (See Figure 4.4)
2. Click the **Animate** button (10).
3. If there is no motion, go to the **Animation** menu and select the item **Start from Beginning** (Ctrl+S) (in the Animator screen).

4. After viewing the animation, exit the animator by clicking the ✕ in the upper-right corner of the screen.

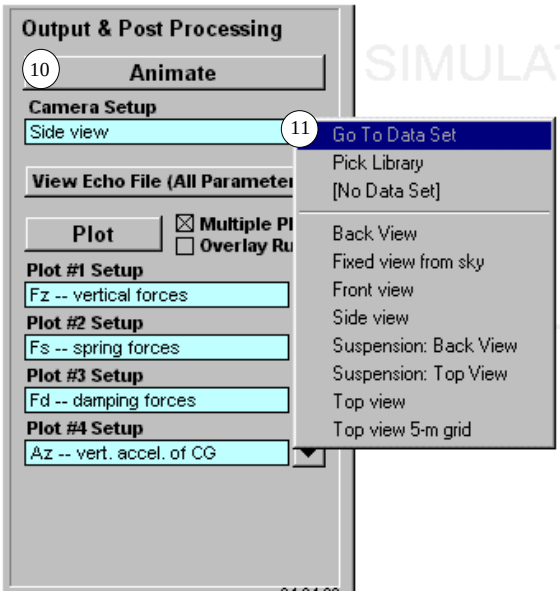


Figure 4.4. Choosing a camera setup for the animator.

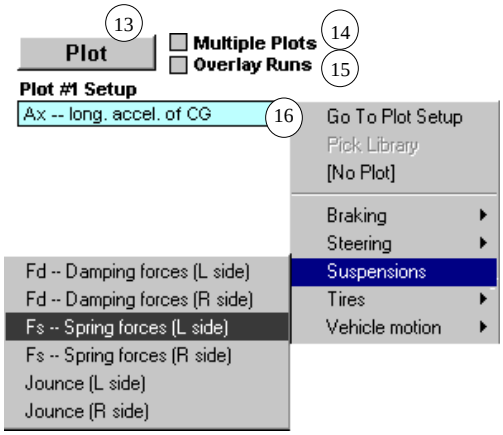
Viewing a Single Pre-Defined Plot

Start from the **Runs** screen (see Figure 4.1 on page 30). (The simulation run must have already been made in order to view a plot.)

1. To avoid making multiple plots, un-check the box **Multiple Plots** (14).

2. To avoid overlaying plots from different runs, un-check the box **Overlay Runs** (15).

3. Use the pull-down menu next to the blue field labeled **Plot #1 Setup** (16) and select the plot of interest (e.g., spring forces on the left side of the vehicle).



4. Click the **Plot** button (13).
5. After viewing the plot, exit the plotter by clicking the in the upper-right corner of the screen.

Viewing Several Pre-Defined Plots

Start from a **Runs** screen (see Figure 4.1 on page 30). (The simulation run must have already been made in order to view a plot.)

1. Check the box **Multiple Plots** (14). This reveals three more links to plotter data sets (17), as shown in Figure 4.5.

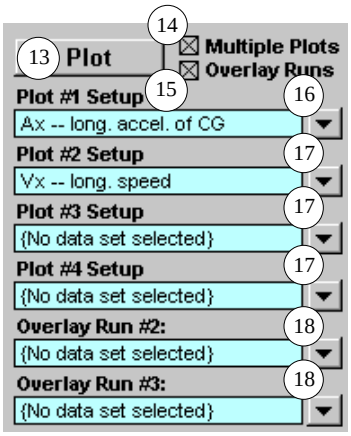


Figure 4.5. Additional plot and run links.

2. Select up to four plot setups, using the links (16) and (17).
3. Click the **Plot** button (13).

> Note: Multiple plots can be viewed in WinEP in several modes. The WinEP **Windows** menu can be used to locate any existing plot window. All of the plots can be viewed simultaneously by using the vertical or horizontal tiling options from the **Windows** menu.

4. After viewing the plots, exit the plotter by clicking the in the upper-right corner of the screen or by using the **File** menu or by typing Ctrl+Q.

Overlaying Plots for Multiple Runs

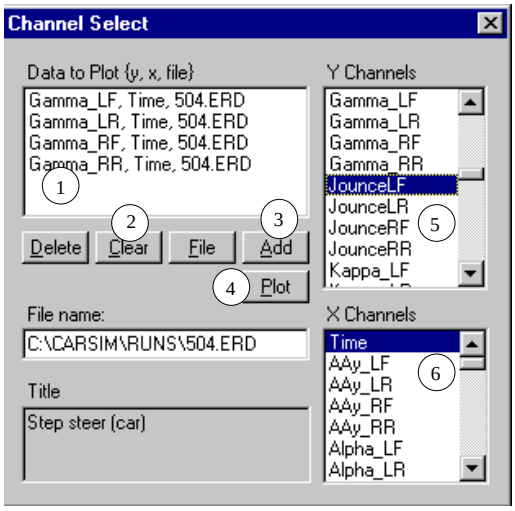
Start from a **Runs** screen (see Figure 4.1 on page 30). Multiple simulation runs must have already been made in order to view plots involving them.

1. Check the box **Overlay Runs** (15). This reveals two links labeled **Overlay Run #2** and **Overlay Run #3** (18) (see Figure 4.5).
2. Select another run of interest from the pull-down menu next to **Overlay Run #2** (18).
3. If you want to overlay three runs, link to another run of interest using the link labeled **Overlay Run #3**. On the other hand, if you only want to compare two runs, choose **{No data set selected}** for **Overlay Run #3**.
4. Select up to four pre-defined plots using links (16) and (17).
5. Click the **Plot** button (13).
6. After viewing the plots, exit the plotter by clicking the in the upper-right corner of the screen or by using the Exit option in the **File** menu or by typing Ctrl+Q.

Defining Plots Interactively

Start from a **Runs** screen (see Figure 4.1 on page 30). (The simulation run must have already been made in order to view a plot.)

1. Start the plotter program by clicking the **Plot** button. (See (13) in Figure 4.1.)
2. From within the plotter, go to the **Data** menu and select the item **Define New Plot…** (or type the keyboard command Ctrl+N). This brings up a window which lists all of the variables in the simulation output file.



3. Select X and Y variable names from the X and Y lists (5) and (6).

> Note: You can select the names by clicking on them with the mouse. You can also click in a list to make it active and then type the first letter of a name. As with other lists in Windows, keep hitting the keyboard letter to go down the list. For example, hit

"J" three times to select the third variable that begins with the letter "J".

4. Add the selected variables to the list of data to plot (1) by using the **Add** button (3).

Note: You can also add a pair of variables by double-clicking on the name in either the X (5) or Y (6) lists. You can also type Alt-A.

4. Make the plot by clicking the **Plot** button (4).

5. To make another plot, repeat steps 2 through 4. To clear the list of the data to plot use the **Clear** button (2).

See Chapter 7 for more information on using the plotter interactively.

Viewing All Model Parameters and Inputs

Start from a **Runs** screen (see Figure 4.1 on page 30). (The simulation run must have already been made in order to view a list of all parameters.)

1. Click the button **View Echo File (All Parameters)** (12).

2. This loads a text file into the default text editor. The text file is created when a solver program is run, and lists all model parameter values, definitions, and units.

Notes: CarSimEd is shipped with a free text editor called WinVI. If you want to use a different text editor as your default, see the section **Assigning the Default Text Editor** in Chapter 10.

3. Return to the **Runs** screen in CarSimEd with the text editor still active by pressing `Alt+Tab`. Or, exit the text editor to return to the **Runs** screen.

Printing a Data Set

You can print any screen in CarSimEd. Copies of the data screens are sometimes helpful for showing model parameters as they appear when using the software. However, a more efficient method is to view all of the parameters in a text editor, as described in the previous section, and then print them from the text editor.

Start from a **Runs** screen (see Figure 4.1 on page 30). (The simulation run must have already been made in order to view a list of all parameters.)

1. Click the button **View Echo File (All Parameters)** (12).

2. Select **Print** from the **File** menu of the text editor.

3. Exit the text editor.

Making a New Vehicle Data Set

Use the following method to create a new vehicle data set that can be used for future runs. Using this process leaves existing vehicle descriptions intact.

The parameter values for a vehicle model are spread over several data sets. Typically, you will want to change a few and leave most of the rest the way they are.

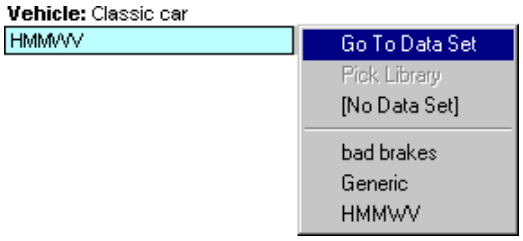
> Note: In the following instructions, the concept of going "up" and "down" is based on the map shown in Figure 3.4 on page 26.

Method

Start from a **Runs** screen (see Figure 4.1 on page 30).

1. Find an existing data set that involves the type of vehicle you are interested in. To navigate through the **Runs** library, use the buttons (2) next to the Data Set (title) box (1).

2. Go down to the linked vehicle data set.



3. Repeat step 2 as needed to go down to more detailed levels. To start, go down to the linked data set for a vehicle. If you are interested in making a new component or subsystem data set (tire, suspension, etc.), use the appropriate link to go down to it. As indicated in the map shown in Figure 3.4 on page 26, it is possible to go down through several levels of detail.

For the purpose of making changes, you are at the "bottom" when

- the current screen has no blue links, or
- the current screen has blue links, but the data sets available from the menus associated with the links are suitable for your purposes. In other words, you do not have to modify or create any data sets that are further "down" in the database.

4. Click the **New** button (in the ribbon bar) to copy the current data set.

5. Give the new data set an appropriate title in the yellow "Data Set" field (1).

6. Modify the data set:

- Modify the values in the yellow fields as needed.
- Change the data sets in any blue links, using the adjacent pull-down menu.

> Note: When you clicked the **New** button, you created a new data set that is now available for use in CarSimEd. It is in the same library as the original, and appears in pull-down menus that show the library contents. However, it will not be used in any simulations until you have linked to it from a higher-level data set.

7. Go back up one level, using the **Back** button in the ribbon bar.

8. Repeat steps 4 through 6. In this case, the change you will make is to select the new data set you created in the library one level "down." (Use the blue link that connects to the library in which you just added a data set.)

9. Repeat steps 7 and 8 until you are back at the **Runs** screen. At this point, you can make a new run or modify the current run. Either way, you will select the new vehicle data set from the linked library of vehicles.

Example

Suppose you are going to change the spring in the front suspension of the car. Starting from the **Runs** screen, you would do the following:

a. Go down to the vehicle data set.

b. Go down to the front suspension data set.

c. Make a new suspension data set, and then change the spring rate.

d. Return to the vehicle data set. Make a new one, and change the suspension link to use the data set created in step c.

e. Return to the Runs data set. Make a new one, and change the vehicle link to use the data set created in step d.

f. Click the Run Simulation button.

Modifying an Existing Vehicle Description

Use the following method to modify an existing vehicle description. Start from a **Runs** screen (see Figure 4.1 on page 30).

1. Find an existing data set that involves the vehicle whose description you will modify.

2. Go down to the linked vehicle data set. Change any yellow fields or blue links as needed.

3. Repeat step 2 to go down more links and change other components as needed.

4. When all changes have been made, return to the **Runs** screen. If you are working with the 3D handling model, return to the **Runs** screen by clicking the Runs button in the ribbon bar ( ) or using the **Runs** command in the **Tools** menu. However, to return to the **Runs** screen for the ride model or suspension kinematics, you must use the **GO** button (or click the **Back** button repeatedly to retrace your steps back to the **Startup** screen).

> **Warning:** Almost every data set in the CarSimEd database is referenced by other data sets (via the blue links). When you change values in a data set, you have in effect also changed all data sets that reference it. For example, if you change a spring rate, every vehicle data set that is linked to the affected suspension data set with will use the new spring rate in future simulation runs.
>
> Hopefully, this is exactly what you intend. If not, then consider using the **New** button to create a copy of the data set, then change the copy and link to it in those places where you want to use the new data.

Switching Between CarSimEd and SIMULINK Inputs

When you run the CarSimEd models from SIMULINK, all of the inputs in the database can be used. Additionally, inputs can be provided from the SIMULINK environment.

The control inputs for the model are the sums of the inputs from the CarSimEd database (the simulation graphical user interface—the SGUI) and those from the SIMULINK work space. The SIMULINK inputs are all elements in an array and are typically referenced in SIMULINK by the array index. Each folder for a SIMULINK model includes a copy of the README text file that defines the inputs for the 3D vehicle model, and tells how they are combined with inputs from the SGUI. (For example, Appendix __ lists part of the README file.)

Given that each model input is the sum of two parts—one from the SGUI and one from SIMULINK—one or both of the components are typically set to zero. If both are zero, the input to the model is zero. If the SGUI input is zero, then all control comes from SIMULINK. And if the SIMULINK input is zero, then all control comes from the SGUI.

To Specify Inputs from the CarSimEd Libraries (SGUI)

1. Specify inputs as described earlier for making runs with the stand-alone solver programs. For example, choose an open-loop steering input that defines steering wheel angle as a function of time.

2. Attach the associated inputs to ground in SIMULINK (in this context, “ground” is the electrical concept that sets a variable to zero). See the README file in the `Matlab`

folder to identify the input variables. For example, connect the steering wheel input contribution [INPT(1)] to ground. (Or, just don't specify it at all because the default value is zero.)

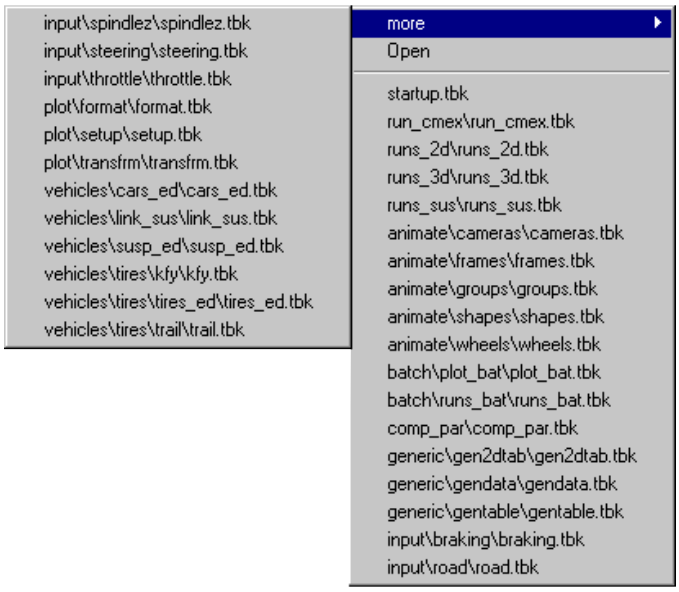
To Specify Inputs from SIMULINK

1. Select inputs as described earlier for making runs with the stand-alone solver programs. For example, choose the "No Steering" data set for the steering wheel angle as a function of time. The control inputs are all tabular functions of time, so just specify data sets where the input variable is zero for all values of time.
2. Specify the associated inputs in SIMULINK using the SIMULINK and MATLAB modeling capabilities. See the README file in the `Matlab` folder to identify the input variables. For example, specify a mathematical function for the steering wheel input contribution [INPT(1)].

Going Directly to Any CarSimEd Library

You can jump directly from any CarSimEd library to any other library.

1. Press the **GO** button in the ribbon bar to display a menu of other libraries in CarSimEd. Select a library from this menu to leave the current library and go to the selected one.



CarSimEd menus

The menu typically contains more items than will fit on a monitor with VGA resolution (640 x 480). In this case, the first item in the menu is **More**. Highlight it to view more menu items.

Notes: After using the GO menu, you can return to the previous library using the **Back** button from the ribbon bar.

When you go to a library using the GO menu, CarSimEd shows you the most recently viewed data set. This data set is not necessarily one that is used in the most recently made run.

Different menus are created for the GO button when you start the software, depending on the simulation type. For example, when you start with the 3D suspension simulation, most of the vehicle data sets are not shown in the GO menu because they cannot be used with the suspension model.

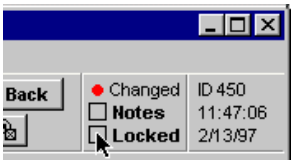
To run with a GO menu that includes all data sets in CarSimEd, start with the data set **Install** (in the Startup library).

Locking Your Data

CarSimEd is designed to make it easy to answer "What if?" questions. Changing vehicle parts, inputs, run conditions, etc. is simple and rapid. However, you might overwrite some data you meant to keep, and you will no doubt conclude that changing parameters is too easy! CarSimEd does have a means to protect your data from inadvertent changes.

Protecting a Single Data Set

1. Check the **Locked** box in the upper-right corner of a data screen.



Click here

Notes: When the locked box is checked, try modifying a blue link or a yellow field. You will find that all pull-down menus next to blue links allow only the navigation option to go to the currently selected data set. You will also find that clicking on a yellow field brings up a message telling you that the data set is locked.

At any time, you can unlock the data set by clicking the box again to un-check it.

The locked box protects only the data shown on the screen. Data sets in linked libraries are not affected. For example, you might lock a **Runs** data set. However, you could still go to the vehicle data linked to the run and change the vehicle data set if it is not locked.

Locking Multiple Data Sets All At Once

1. From any data screen in the library of interest, click the button in the ribbon bar or select the **Library Editor** command from the **Tools** menu. This brings up the floating palette shown in Figure 4.6.
2. Select the data sets to be locked from the list in the dialog box.
3. Click the **Lock Data Sets** button.
4. Click the upper-left corner of the floating palette window to close it.

> Note: Multiple data sets can be unlocked using the same method, except that you click the button **Unlock Data Sets.**

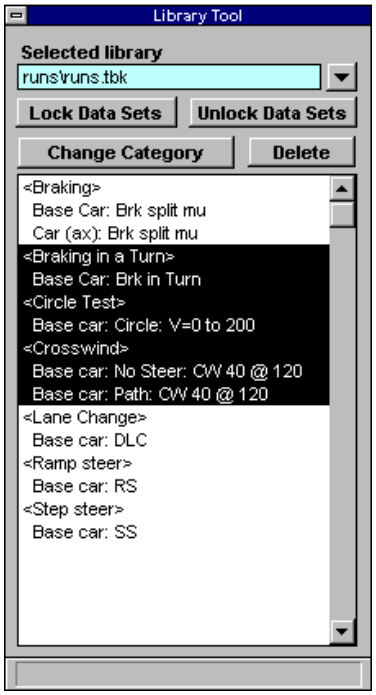


Figure 4.6. Library Editor.

Automatically Locking All Data Sets As You Create Them

1. From any CarSimEd data screen, click the button in the ribbon bar or select the **Tools** menu **Preferences** command. This brings up a dialog box with a few check boxes, as shown in Figure 4.7.

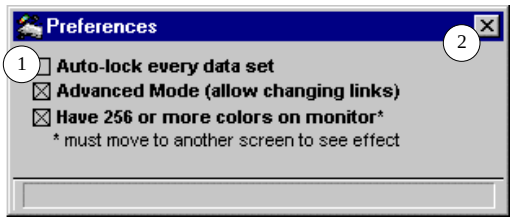

Preferences
1
2
Auto-lock every data set
Advanced Mode (allow changing links)
Have 256 or more colors on monitor*
* must move to another screen to see effect

Figure 4.7. Preferences window.

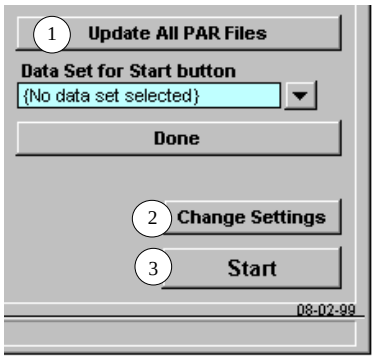
2. Check the box **Auto-lock every data set** (1).
3. Close the dialog box by clicking the ☒ (2) in the upper-right corner.

> Notes: When the **Auto-Lock** feature is enabled, every data set that you view will be locked automatically when you leave it.
>
> This option does not change the status of any existing data sets. They will be locked after you visit them.
>
> This feature can be disabled at any time by returning to the preferences screen and un-checking this box.

Automatically Locking All Data Sets in CarSimEd

You can extend the above method to cause CarSimEd to lock all data sets.

1. Follow steps 1 through 3 above.
2. Use the **GO** button in the ribbon bar to go to the CarSimEd startup screen (`Startup.tbk`).
3. Go to the data set named `Install` (this is a data set included in the software as installed).
4. Click the button **Change Settings** (2). This will hide the CarSimEd logo and reveal more buttons, including the ones shown here.



4. Click the button **Update All PAR Files** (1). Besides updating all PAR files, the **Locked** buttons on every screen will be checked.

5. After the update is complete, return to the **Runs** screen by clicking the **Start** button (3) or by using the GO button.

Deleting Data Sets

CarSimEd has several methods for deleting data sets.

To Delete the Current Data Set

1. Click the **Delete** button in the ribbon bar. CarSimEd will display a warning, asking you to confirm that you want to delete the data.

> Notes: You can by-pass the warning message by holding the Control key down when you click the **Delete** button.
>
> The **Delete** button cannot be used if the **Locked** button is checked. The data set must be unlocked before you can delete it.

To Delete All Data Sets Except the Current One

1. Click the **Delete** button in the ribbon bar while holding down the Shift key. CarSimEd will display a warning, asking you whether you want to cancel, delete all (including locked sets), or only the unlocked data sets.

To Delete an Arbitrary Group of Data Sets

1. Click the button in the ribbon bar or select the **Tools** menu item **Library Editor**. This brings up the floating palette shown in Figure 4.6. It has controls for selecting data sets by their titles, and a button for deleting the selected data sets all at once.

Making Five or More Plots for a Single Run

The **Runs** screen has links to set up four plots. If you want to quickly generate five or more plots, you can use the **Plot Setup: Batch** screen, shown in Figure 4.8.

Start from any CarSimEd data screen.

1. Click the button (16) in the ribbon bar or use the **Batch Plotting** command from the **Tools** menu. This takes you to the **Plot Setup: Batch** library.

2. Go to a data set in the library that is linked to the current **Runs** library. (As installed, the library contains a single data set linked to the CarSimEd **Runs** library.)

3. Click the **New** button (15) to create a new data set for your new combination of plots and runs.

4. Using the **Add** (3) and **Remove** (4) buttons for the **Data Files**, put the name of the run of interest into the field with **Selected Data Files** (6).

5. Using the **Add** (9) and **Remove** (10) buttons for the **Plots**, put the names of all **Plot Setups** of interest into the field named **Selected Plots** (11).

6. Click the **Make Plots** button (14). This starts the program WinEP and instructs it to make one plot for each item listed in the field **Selected Plots** (11).

7. After viewing the plots, exit the plotter.

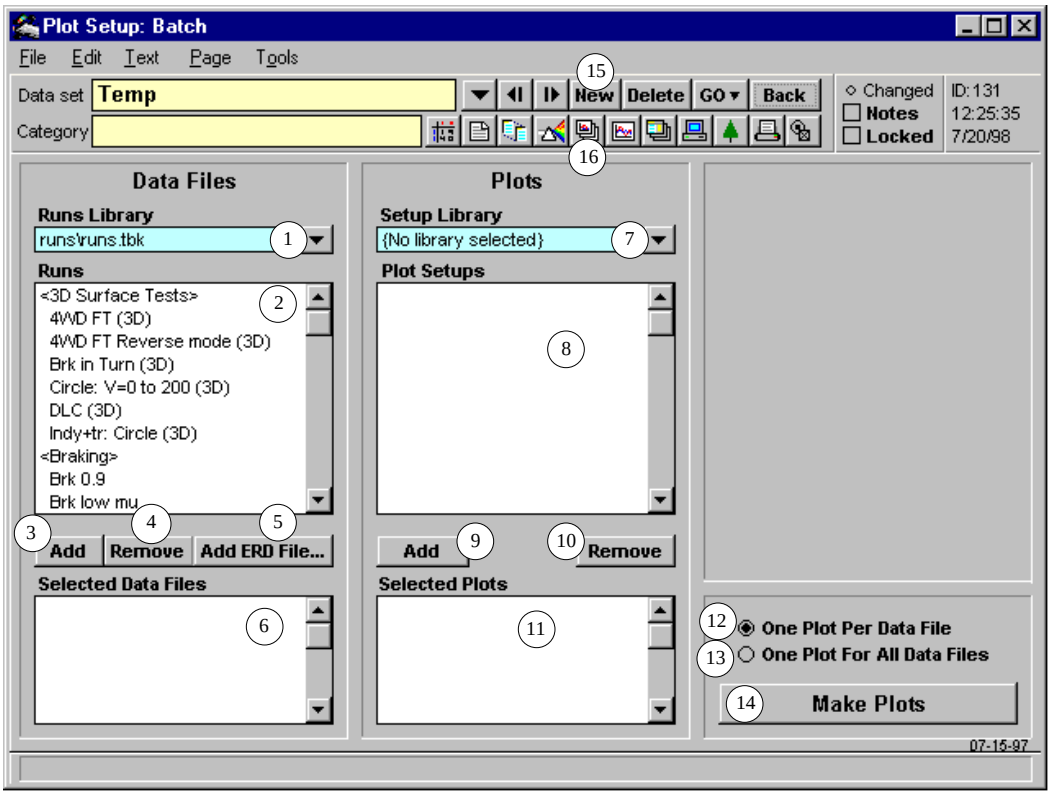


Figure 4.8. Setup for batch plotting.

Overlaying Plots for Multiple Runs

The **Runs** screen has links to overlay data from three runs. If you want to overlay data from more runs, you can use the **Plot Setup: Batch** screen, shown in Figure 4.8.

The method is nearly identical to the one just described for making five or more plots for a single run. Two additional steps are:

- In step **4**, put more than one run into the field: **Selected Data Files** (6).

- Be sure the **One Plot for All Data Files** button (13) is selected.

Making Many Plots in Batch Mode

You can generate a standard set of plots for many runs. Again, you will use the **Plot Setup: Batch** screen, shown in Figure 4.8.

The method is nearly identical to the one just described for making five or more plots. Two additional steps are:

- In step **4**, put more than one run into the field with **Selected Data Files** (6).
- Be sure the **One Plot per Data File** button (12) is selected.

Setting Up Multiple Runs

It is possible to set up more than one run ahead of time. It is also possible to re-do a set of runs whose inputs have been modified. To do this, start from the **Runs** library.

> Note: The batch option is designed for stand-alone (EXE) solver programs. Batch operation for SIMULINK cannot be controlled from the CarSimEd database.

1. Set up the runs of interest by creating a distinct **Runs** data set for each combination of vehicle, input, speed, stop time, etc. that is of interest. Do not click the **Run Simulation** button.
2. Go to the **Runs: Batch** library, whose screen is shown in Figure 4.9. Get there by clicking the button (9) in the ribbon bar or use the **Tools** menu item **Batch Runs**.
3. Click the **New** button (10) to create a new data set for your combination of runs.
4. Using the **Add** (3) and **Remove** (4) buttons, put the name of the runs of interest into the field named **Data Sets to Run** (5).
5. Optionally enter parameters (with keywords) in the yellow fields (6) to override the parameters that are specified in the selected **Data Sets to Run**. For example to make all runs with a simulation speed of 100 km/h, enter the line:

   ```
   speed 100
   ```

> Note: See Appendix F for a complete list of keywords recognized by the solver programs in CarSimEd.

6. Optionally use Links 1, 2, and 3 (7) to link to data sets that will override those specified in the selected runs. For example, to make all the runs with no braking input, link to the braking input library and select the data set **No Braking**.

7. Click the **Make Runs** button (8) to make the batch of runs.

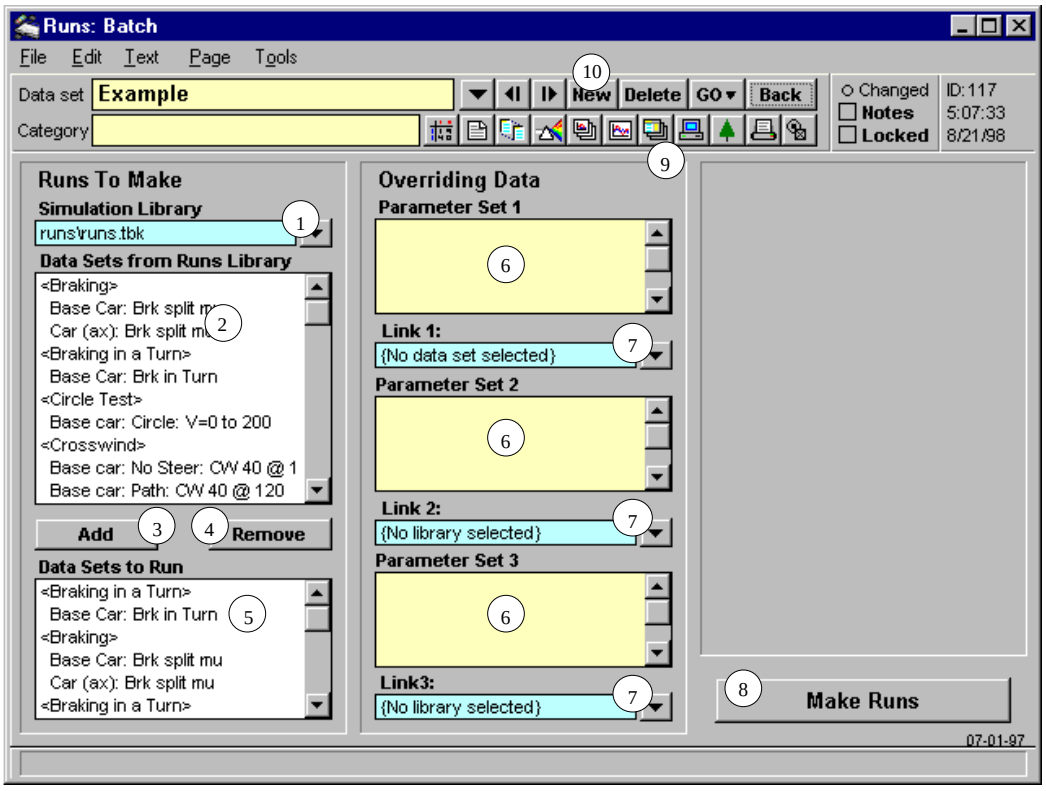


Figure 4.9. Batch run screen.

Notes: You can view the results from the individual runs data sets, or use the batch plot library.

If you specified any parameters or links in steps 5 and 6, then the individual run data sets are modified such that the link with the label **Overriding Data (from Batch)** is set to the batch data set. That way, you can go back to this data set (batch run) to see what parameters or links were applied when the run was made.

The echo files generated for each run show the model parameters that were used in the run, regardless of whether they came from the normal **Runs** data set or from the batch override.

Making New Plot Descriptions

CarSimEd comes with about 30 plot descriptions. It's easy to make more.

Start with any data screen in CarSimEd.

1. Go to the **Plot Setup: Single** library, whose screen is shown in Figure 4.10. Get there by clicking the button (5) in the ribbon bar or select the **Tools** menu item **Plot Setup**.

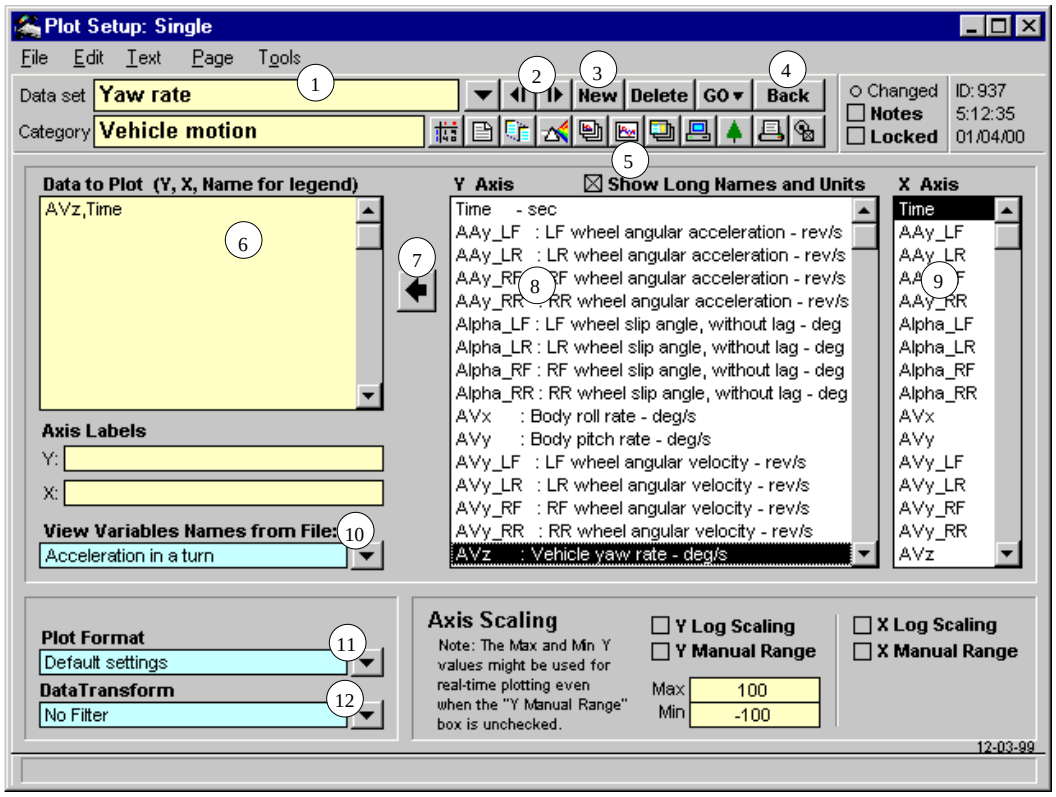

Plot Setup: Single
File Edit Text Page Tools
Data set Yaw rate
Category Vehicle motion
New Delete GO Back
Changed Notes Locked
ID: 937 5:12:35 01/04/00
Data to Plot (Y, X, Name for legend)
AVz,Time
Y Axis
Show Long Names and Units
X Axis
Time - sec
AAy_LF : LF wheel angular acceleration - rev/s
AAy_LR : LR wheel angular acceleration - rev/s
Alpha_LF : LF wheel slip angle, without lag - deg
Alpha_LR : LR wheel slip angle, without lag - deg
Alpha_RF : RF wheel slip angle, without lag - deg
Alpha_RR : RR wheel slip angle, without lag - deg
AVx : Body roll rate - deg/s
AVy : Body pitch rate - deg/s
AVy_LF : LF wheel angular velocity - rev/s
AVy_LR : LR wheel angular velocity - rev/s
AVy_RF : RF wheel angular velocity - rev/s
AVy_RR : RR wheel angular velocity - rev/s
AVz : Vehicle yaw rate - deg/s
Axis Labels
Y:
X:
View Variables Names from File:
Acceleration in a turn
Plot Format
Default settings
DataTransform
No Filter
Axis Scaling
Note: The Max and Min Y values might be used for real-time plotting even when the "Y Manual Range" box is unchecked.
Y Log Scaling
Y Manual Range
Max 100
Min -100
X Log Scaling
X Manual Range
12-03-99

Figure 4.10. The Plot Setup screen.

2. Use the navigation buttons (2) in the ribbon bar to find a plot setup similar to what you want.

3. Click the **New** button (3) to make a new data set.

4. Type a name for the new data set in the title field (1). This name will later appear when you view the list of plots with the plot setup menus on the **Runs** screen.

5. Select a run from the library linked to the field named **View Variable Names from File** (10). When you select a run, the output ERD file associated with the run is scanned and all of the variables are listed in fields (8) and (9)**. The run must have been made in order for the ERD file to exist.**

Notes: It doesn't matter which run you use to build the file description, so long as the ERD file contains the variables that you want

plotted. The only reason an existing run is used to build the description is because it is more convenient to click on names in a list rather than typing in the names.

(However, if you prefer to type, you can type the names directly into the yellow field (6).)

If the ERD file has more than about 150 variables, only the short names are shown in fields (8) and (9). If it has fewer variables, more information is shown.

6. Clear the field called **Data to Plot** (6). (Select the contents of the field and then press the delete key.)
7. Select X and Y channels of interest from fields (8) and (9). Add them to the list of **data to plot** (6) using the button (7) or by double-clicking in either field (8) or (9).
8. Continue to build the list of data to plot (6).
9. To make another plot setup, repeat steps 2 through 8.
10. When you are through making new plot descriptions, leave the library by clicking the **Back** button (4) or using the GO button.
11. You can now plot the new plot using the same method as for the standard plots.

Linking to Different Libraries

As installed, all blue links are preset to the appropriate libraries. In general, when you want to make a new data set, you should find an existing data set that has links in place to the intended libraries. However, the CarSimEd database has a provision for changing the linked libraries as needed.

1. On any screen, select the button next to the link of interest to display the pull-down menu. If the second item **Pick Library** is active, select it. However, if it is dimmed, you must first change a global setting in CarSimEd.
    a. Go to the preferences window by clicking the button in the ribbon bar or selecting the **Preferences** command from the **Tools** menu.
    b. Make sure the **Advanced Mode (allow changing links)** box is checked and then close the preferences window.
    c. Try step 1 again.
2. When you select the menu item Pick Library, a Windows File Select dialog box appears. Browse your file system to locate the intended TBK file and select it.

Note: If you are not sure where the library of interest is located, you can refer to the description in Chapter 9, which will include the pathname to the TBK file. Alternatively, you can display the GO menu (see page 44) and view the relative path name of the TBK file of interest from the menu.

3. After linking to a library, the blue field will read {No data set selected}. Although you have linked to the library, you still have to link to a data set within that library. You do this with the same pull-down menu, which will now display the names of the data sets in the newly linked library. Simply pull down the menu and pick a data set to link to.

5. The Solver Programs

CarSimEd contains programs that solve the equations of motion for vehicle models, predicting motions, forces, and other output variables. A solver program is run automatically whenever you click the **Run Simulation** button from the **Runs** screen. CarSimEd allows you to set up inputs to the programs and view their outputs without ever dealing with the details of how these programs work. However, it can sometimes be useful to understand how they read and write files.

Overview of Program Operation (Stand-Alone)

The solver programs are so-called "console applications." Each behaves as a terminal console, with no user interface other than the display of text and the acceptance of keyboard entries. They have the same appearance as a plain DOS text program, although they are technically 32-bit Windows programs (see Figure 5.1).

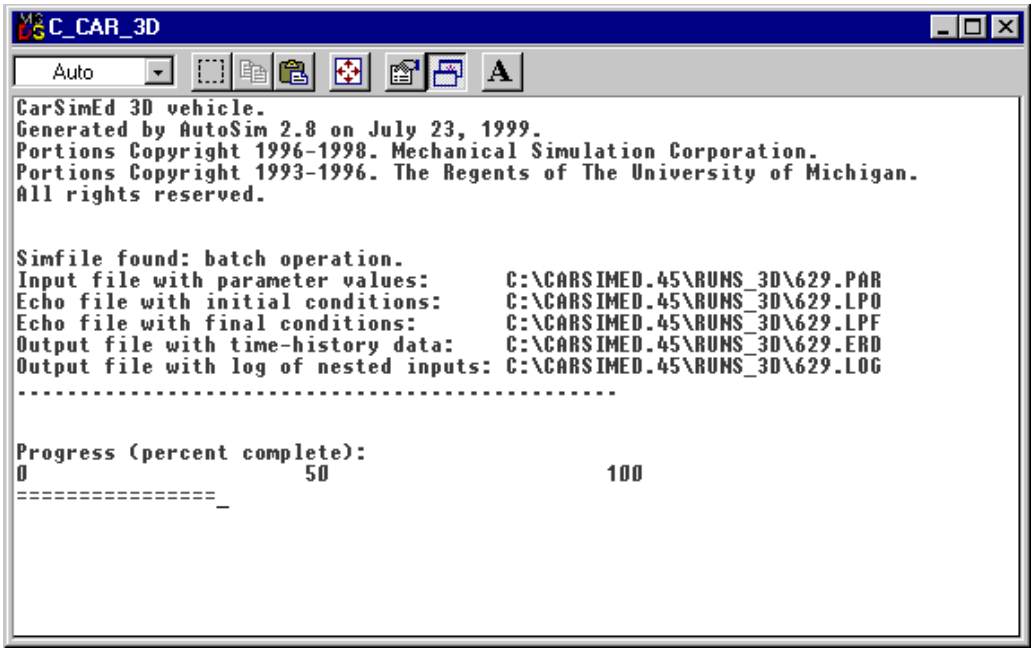

C_CAR_3D
Auto
CarSimEd 3D vehicle.
Generated by AutoSim 2.8 on July 23, 1999.
Portions Copyright 1996-1998. Mechanical Simulation Corporation.
Portions Copyright 1993-1996. The Regents of The University of Michigan.
All rights reserved.
Simfile found: batch operation.
Input file with parameter values: C:\CARSIMED.45\RUNS_3D\629.PAR
Echo file with initial conditions: C:\CARSIMED.45\RUNS_3D\629.LPO
Echo file with final conditions: C:\CARSIMED.45\RUNS_3D\629.LPF
Output file with time-history data: C:\CARSIMED.45\RUNS_3D\629.ERD
Output file with log of nested inputs: C:\CARSIMED.45\RUNS_3D\629.LOG
Progress (percent complete):
0 50 100

Figure 5.1. Screen display when solver program is running.

When the program starts, it creates a window in which text is displayed. The name of the program file is typically shown as the window title.

The solver programs can be run interactively or in batch mode. The mode is determined by the existence of a batch control file with the name `Simfile`, a file created

automatically by CarSimEd: if this file exists, the program runs in batch mode. Otherwise, it runs interactively.

> Note: All normal runs in CarSimEd are performed in batch mode. CarSimEd creates `Simfile` and then launches the solver program that matches the type of vehicle selected on the **Runs** screen.

If `Simfile` does not exist in the folder in which the program is located then the program runs in interactive mode. It prompts you for an input file, an output ERD file name, and two output echo file names. After the last file is written, the program quits.

If `Simfile` does exist, then all file names are read from it. The program quits when `Simfile` is fully processed and the solver program writes the last output file.

Properties of the window created by the solver program can be adjusted by right-clicking while the cursor is positioned over the solver program (extension EXE, found in the folder `CarSimEd\Programs`) to display a pop-up menu, and selecting **Properties** from the menu.

As installed, the solver programs close their windows when they finish. When the window is closed, the most recently accessed program comes to the top. When run normally from within CarSimEd, the most recent program is ToolBook, which was showing the **Runs** screen from which the run was initiated. Therefore, the normal behavior is that the Solver program brings up a window, runs for the time needed to perform the simulation calculations, then quits and closes the window, leaving you back at the **Runs** screen.

How a Simulation Run Is Made in SIMULINK

Simulation runs can be made by opening a SIMULINK model that includes the CMEX S-function, and then initiating the run from within SIMULINK.

For example, Figure 5.2 shows a simple model involving a CarSimEd S-function called `is_cmx`. The SIMULINK model resembles the flow chart shown in Figure 3.7 on page 29, except that the SGUI input and ERD file outputs are not visible (they are part of the CMEX code and cannot be modified in SIMULINK).

The example model in Figure 5.2 routes a few of the outputs to scopes to generate the time history plots shown.

When a simulation run is made, the text that is shown in the program window for a stand-alone EXE CarSimEd program instead appears in the MATLAB window, visible in the background in Figure 5.2.

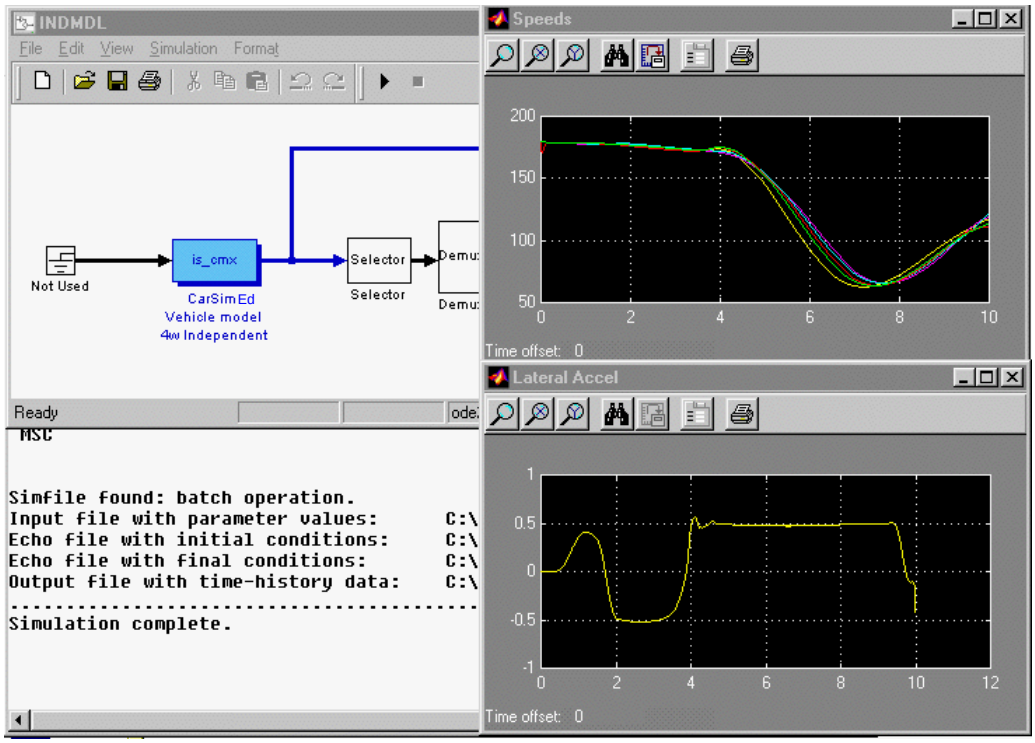


Figure 5.2. A SIMULINK model with a CarSimEd CMEX S-function.

Starting SIMULINK from within CarSimEd

When the first run is made in a CarSimEd session, it is necessary to launch MATLAB/SIMULINK and then make the run. As a user, you click the button **Start SIMULINK** on the **Runs_cmx** screen to launch SIMULINK. CarSimEd initiates the sequence shown in Figure 5.3 and described below.

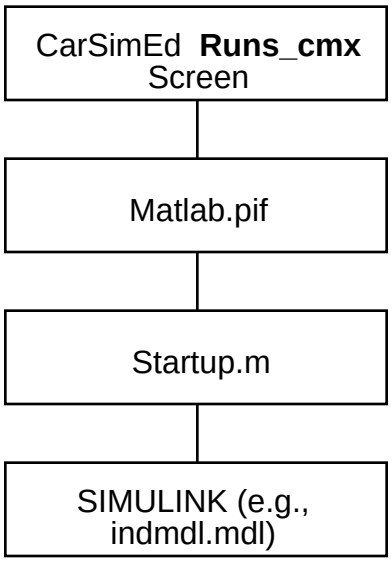


Figure 5.3. Transfer from CarSimEd to SIMULINK.

1. For each SIMULINK model (ABS controllers, etc.) there must be an independent PIF in its own folder.

   When you click the **Start SIMULINK** button, CarSimEd creates a SIMFILE in the same folder that contains the PIF. For example, the PIF pathname might be `C:\CarSimEd.45\Matlab\is_cmex\Matlab.pif`.

2. CarSimEd then sends a run command to the PIF.

3. The PIF is linked to your copy of MATLAB (see Figure 2.2 on page 19). It launches MATLAB with the current working directory set to the folder containing the PIF, e.g., `C:\CarSimEd.45\Matlab\is_cmex`.

4. When MATLAB starts, it tries to run the command `Startup`. To do so, it looks for a file in the working directory called `Startup.m`. If found, it automatically loads the file and runs it. CarSimEd has a file with a single line that is the name of a SIMULINK model. For example, the file `C:\CarSimEd.45\Matlab \is_cmex\Startup.m` might have the single line:

   ```
   indmdl
   ```

5. The file name mentioned in `Startup.m` is opened. For example, `indmdl` will cause MATLAB to open the SIMULINK model file `indmdl.mdl`. The SIMULINK model file in turn loads the DLL file `is_cmex.dll` which has the compiled CMEX model in the form of an S-function.

When this process is completed (in a few seconds), MATLAB is running with a SIMULINK model in the foreground. Start the run using the SIMULINK **Start** command from the **Simulation** menu. The CarSimEd S-function will read data from the CarSimEd database, using information contained in the SIMFILE.

The integration method, the start time, and the time step are all obtained from SIMULINK. However, the stop time is read from the CarSimEd SGUI.

Making Additional Runs in SIMULINK

After the first run is made, you might want to make more runs using different vehicle parameters. If you change any vehicle properties in the CarSimEd SGUI, set up a new run by clicking the button **Update** on the **Runs** screen to create a new SIMFILE (step 1 above). Then make the new run by selecting the **Start** command from the **Simulation** menu in SIMULINK.

Discussion of SIMULINK/CarSimEd Integration

There are two tricky parts about making a run with CarSimEd and SIMULINK together:

1. The directories used by CarSimEd must be properly communicated to MATLAB.

2. Because the input files from the CarSimEd SGUI are controlled from the CarSimEd Runs screen, and the simulation is controlled from the SIMULINK screen, it is up to you, the user, to assure that inputs are properly updated before a run is made.

With respect to the first item (directories), the PIF is used to set up a working directory without adding more paths to the MATLAB workspace. (Adding pathnames can cause conflicts and uncertainty about which version of a file is loaded.) All files associated with CarSimEd are located in that directory, where the potential for conflict with other MATLAB files is minimal.

With respect to the second item, the main point is that you, the user, must be aware of the role played by CarSimEd and SIMULINK when they are combined. The CarSimEd graphical user interface is used to define parameters and properties of the simulated vehicle. Control inputs can also be specified in the SGUI. However, control inputs are also provided from SIMULINK. (Controls from SIMULINK are added to the inputs from the SGUI.) To choose the source, you should "zero out" the contribution from the unwanted source. To eliminate the input from the CarSimEd SGUI, select zero-valued inputs from the **Runs_cmx** screen.

Any time you change a vehicle property, you must remember to click the button **Update** on the **Runs_cmx** screen to create a new SIMFILE before making a new run in SIMULINK.

File Types

When run from the graphic interface in CarSimEd (i.e., by clicking the **Run Simulation** button from a **Runs** screen), all files associated with a run have the same base name and differ only in their extensions. The base name is the current ID number displayed in the upper-right corner of the **Runs** screen.

Table 5.1 shows the files and their extensions.

Details of the file types are provided in Appendix F, along with some examples. Short summaries are provided below.

Simfile

`Simfile` is the batch control file. Whenever you click the **Run** button, a new `Simfile` is created in the same folder as the solver program (by convention, this is the folder `Programs`). If a file named `Simfile` already exists, it is overwritten. Because `Simfile` is automatically re-generated before each run, it can be deleted at any time without loss of information.

The `Programs` folder contains the Simfile used for the most recent run. Therefore, if you run a solver program independently of CarSimEd, it will repeat the last run made (if it is still in the **Programs** folder).

To run a program in the interactive mode, you must delete or rename `Simfile` or launch the solver program from outside the `Programs` folder.

Table 5.1. Standard files created when a simulation is run.

Name	Location	Creator	Description
`Simfile`	`Programs`	SGUI	Batch control file.
<*id*>`.PAR`	`Runs`	SGUI	Input file with parameter values and links taken from the **Runs** screen.
<*id*>`.LPO`	`Runs`	Program	List of parameters, written as output by program. Also contains initial conditions. Can be used to repeat a run.
<*id*>`.LPF`	`Runs`	Program	List of parameters and final conditions, written as output by program. Can be used to continue a run.
<*id*>`.ERD`	`Runs`	Program	Header for ERD file, needed by plotter and animator programs.
<*id*>`.BIN`	`Runs`	Program	Numerical values of output variables, stored in binary form, and used in association with ERD files.
<*id*>`.LOG`	`Runs`	Program	Log of all input files, including nested PAR files.

PAR Files

The PAR file lists input parameters for the solver program, along with links to other PAR files.

When you click the **Run Simulation** button, the SGUI creates a new PAR file in the folder containing the **Runs** library (e.g., `Runs`). The PAR file contains information from the **Runs** screen such as the title of the run, pathnames for the data sets linked to the run (vehicle, steer input, etc.), and input parameters (e.g., `stopt`)

The PAR file also includes references to other PAR files. Every data screen in CarSimEd has an associated PAR file, including the **Runs** screen. When the solver program runs, it reads the PAR file generated from the **Runs** screen, and all other PAR files referenced within it. The files can be thought of as forming a tree, such as the one shown in Figure 3.4 on page 26.

Viewing the Tree of PAR Files

The actual "tree" associated with any screen in CarSimEd can be viewed by clicking the Parstree icon in the ribbon bar . Clicking the button launches a program called Parstree that displays the PAR file associated with the current data screen. Figure 5.4 shows an example display window from the Parstree program. Note that many of the input lines start with the word `parsfile`, and are followed by pathnames to other files with more data.

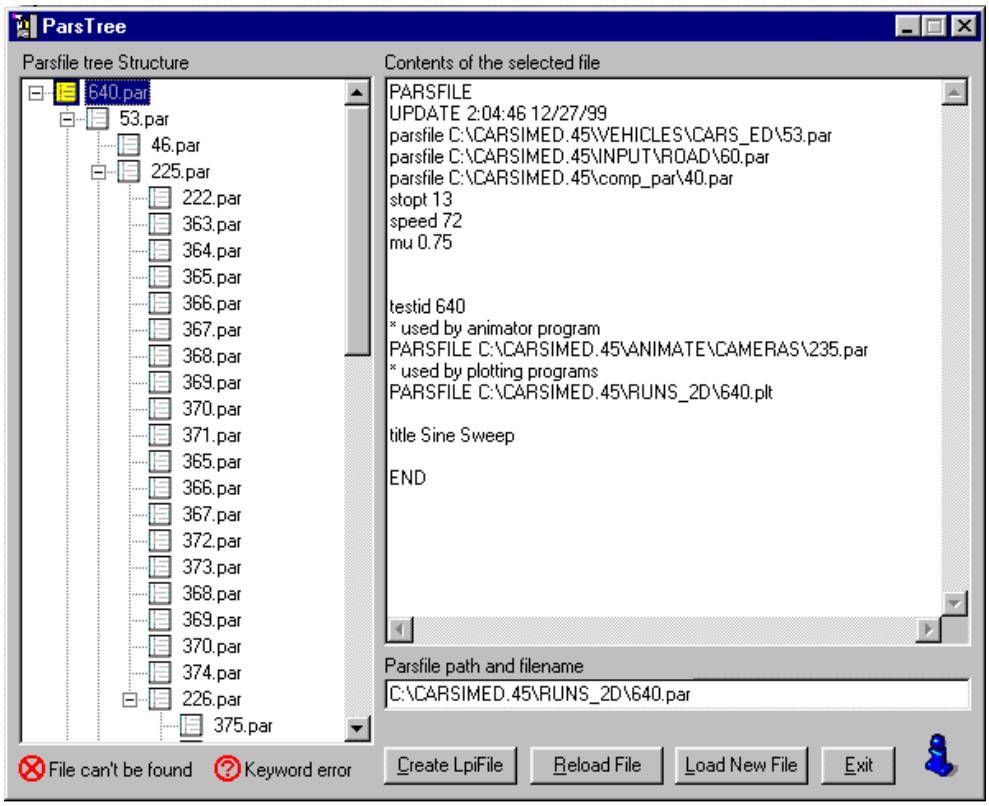


Figure 5.4. An example Parstree for a Runs data set.

LPO and LPF Echo Files

When a solver program runs, it creates summary files that list each and every parameter value. One of these files is created before the run (LPO), and the other is created at the end of the run (LPF). Both files contain all parameter values for a simulation. In addition, the LPO file contains the initial conditions for the state variables in the simulation. The information in the LPO file is sufficient to exactly repeat a run.

The difference between the PAR and LPO files is that the LPO file contains every parameter value in one file (see Appendix F for an example), whereas the PAR file contains pathnames of other files that contain the data (see Figure 5.4). Also, the PAR file may not reference every parameter used by the solver program. For example, the initial conditions listed in the LPO file are normally not specified on input (they are assigned default values by the solver program).

The LPF file is nearly identical to the LPO file, except that instead of initial values, it contains final values of the state variables. By modifying the start and stop times, an existing run can be continued (restarted).

ERD and BIN Files

The main purpose of each solver program is to calculate time histories of variables of interest. Those time histories are stored in a binary data file with the extension BIN. The BIN files contain numerical data organized by channel number and sample number,

similar to test data recorded on a multi-track recorder. A companion file, with extension ERD, describes the layout of the BIN file and also contains labeling information for each variable. It also contains the information that would normally be put into a log sheet summarizing the data, including text needed for preparing graphical plots of the data.

By itself, a BIN file is useless. It has no structure, and cannot be understood without the layout information contained in the ERD header file.

By convention, ERD and BIN file pairs are simply called ERD files. The name ERD is used because the Engineering Research Division at The University of Michigan Transportation Research Institute (UMTRI) designed the format.

Data processing programs for ERD files obtain most of the information needed from the file itself. For example, the high level of automation in the animator and plotter exists because both were designed to work with ERD files.

The format of an ERD file is described in Appendix C. Appendix J contains a list of all of the variables in the ERD and BIN files for one of the CarSimEd models.

Text Output Files

CarSimEd can be made to produce text-only ERD files. In this case, the header information in the ERD file is followed immediately by the numbers, written in text and there is no BIN file.

You can also use CarSimEd to produce simple text output files, for export to spreadsheet programs and other analysis software.

See the section **Computation Parameters** in Chapter 9 for details of how to generate text output files.

CarSimEd always uses the DOS file extension ERD for simulation output files. When you make a text output file, this might be considered be misleading—the file has plain text and does not follow the ERD format.

Log Files

The input parameter files (extensions = PAR) can reference other PAR files, much like an INCLUDE directive in C or Fortran. As the solver processes the parameter files, it writes the names into a LOG file.

Most users will never have occasion to look at the LOG files. They exist because they can help to debug problems involving parameters not getting through, such as when changes are made to the directory structure of the CarSimEd libraries.

6. The Animator

CarSimEd includes a program for animating wire-frame figures to visualize vehicle motions. The animation is accomplished by drawing images similar to what would be seen with a video camera, and updating the images many times per second to show motion.

Updates

The animator program is used in a number of MSC software packages. The latest version is generally available from the Internet at:

`http://www.trucksim.com/animator/index.html`

Updates of the animator program are "backward compatible." If you download a newer version, simply replace the file `Animator.exe` in the `Animate` folder in CarSimEd.

Overview of Operation

To help understand the animator, imagine that you are aiming a video camera at a vehicle as it moves down the road. Thirty times per second, the camera records an image of the vehicle and ground as seen from your point of view. By playing back those images at the same rate (thirty times per second), you see the vehicle move relative to its surroundings.

If the camera has a zoom lens, you can zoom in to fill the screen with the vehicle. Or, you can zoom out to shrink its apparent size.

What you see depends in part on where you are located. You could be standing on the ground, in which case your position would not change. Or, you might be in another vehicle that moves with the vehicle. In the animator, your location is defined as the "camera point."

Your view is also determined by how you aim the camera. You might have the camera aimed at a fixed point in space. The vehicle would only be visible while it is in field of the camera view. More likely, you would move and aim the camera as needed to keep the vehicle in view. In the animator, the camera is aimed directly at a "look point."

Figure 6.1 shows the basic geometry and the relationships between the camera point, the look point, the 3D system being animated, and the 2D image that is recorded.

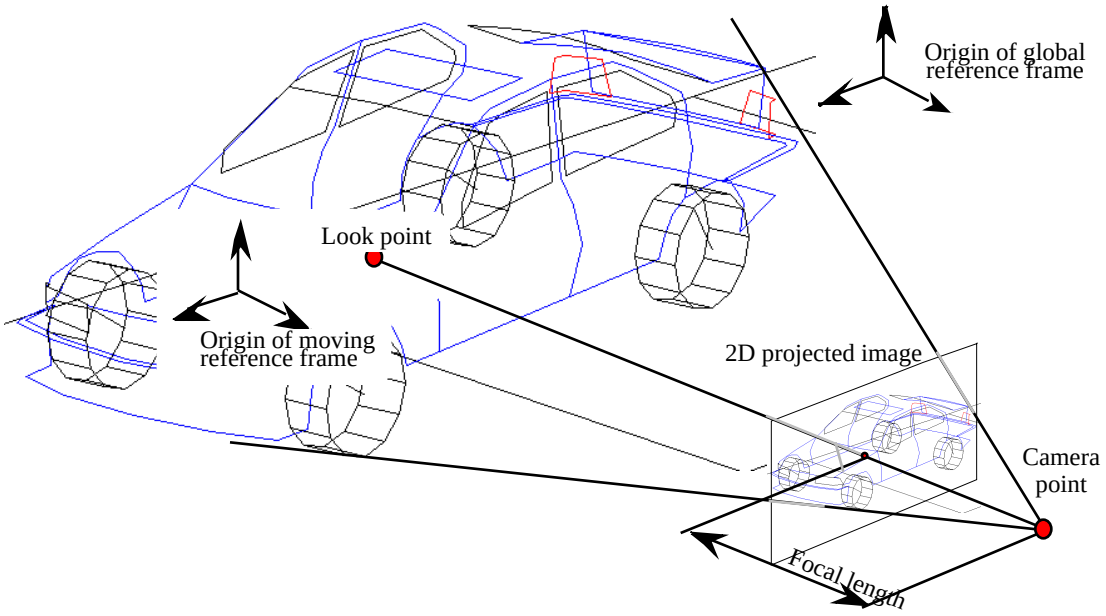


Figure 6.1. Geometry of the camera point and the look point.

The animator allows you, the user, to build and modify descriptions of the system to be animated. In order to use the program effectively, it is helpful to understand the concept of a moving reference frame.

Reference Frames

The animator creates images based on a set of visible objects that includes a grid and arbitrary wire-frame shapes defined by a sequence of connected lines. Some of the wire-frame objects are organized into groups that move together. For example, the body of the vehicle is made up of the bumper, rear bumper, left-front door, etc.

A group of points and objects that maintains a fixed relationship (i.e., that constitute a rigid body) is called a reference frame. Although the reference frame might move and rotate, the spatial relationships between objects in the reference frame do not change relative to each other. In the animator, all motions are associated with reference frames. Each reference frame has a rectangular coordinate system that is used to describe 3D locations of points within that reference frame. For example, Figure 6.1 shows the origins and axes for two reference frames: a global non-moving frame, and a frame that moves with a vehicle body.

In general, it is not possible to tell how many reference frames exist by looking at a single image. However, by observing an animation, it is sometimes possible to see the effects of all reference frames. For example, the system shown in Figure 6.1 includes six reference frames:

- A fixed global reference frame is used to locate a grid that shows the ground plane. The three axes of a coordinate system fixed in this reference frame are shown in the figure.
- A moving frame is associated with the vehicle body. The body is simply a series of lines drawn in this reference frame. The three axes of a coordinate system fixed in this reference frame are shown in the figure. Given that the coordinate system is fixed in a moving reference frame, it follows that the coordinate system moves.
- Each of the four wheels is associated with a separate reference frame.

Wire-frame shapes are all defined using local X-Y-Z coordinates. The animator transforms the local X-Y-Z values to global X-Y-Z values in order to draw the shape. To perform the transformation, the animator must know the location and orientation of the reference frame relative to the global reference frame. This is defined by six variables: three coordinates (global X-Y-Z) and three rotation angles that are called Euler angles.

The six variables needed to locate and orient the reference frame for the vehicle bodies and wheels are all computed as part of the simulation. It is also possible for you, the user, to define new reference frames using combinations of the variables computed in the simulations. Details for specifying reference frames are provided in Chapter 9 in the section **Animator Reference Frames**. That section also provides more information about Euler angles.

Files

When the animator is started from a CarSimEd **Runs** screen, the necessary files are automatically opened. Therefore, it is not necessary to know the names of the input files. However, an understanding of how the animator reads the necessary information from files might be helpful in understanding its operation.

The animator program reads two kinds of input files (see Figure 6.2). As noted above, all motion is due to movements of reference frames, as defined by up to six variables. Those six variables are read from an ERD file that was created by a CarSimEd solver program. Appendix C provides details of the ERD file format, and Appendix J shows a list of variables contained in a typical CarSimEd ERD file.

Other information such as program settings, definitions of parts, shape information, etc. is described with keyword-based text files, typically with the extension PAR. These files, called PARSFILEs, have the same general design as the input files read by the solver programs. A single top-level PARSFILE contains the names of other PARSFILEs with camera information, vehicle information, reference frames, etc.

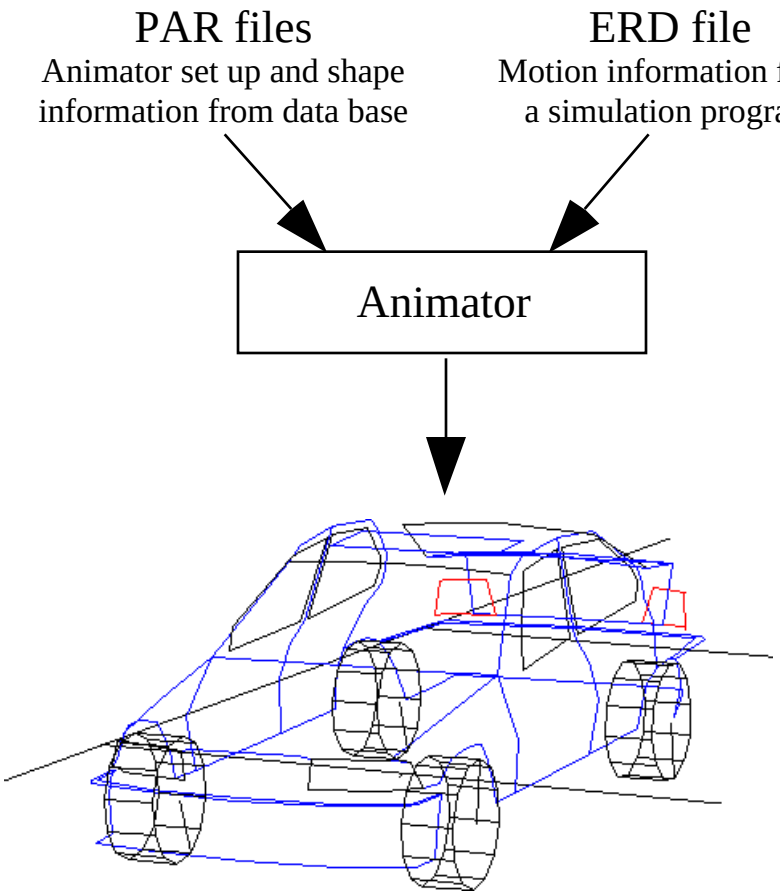

PAR files
Animator set up and shape information from data base
ERD file
Motion information from a simulation program
Animator

Figure 6.2. Animator input files.

Appendix E describes the keywords used in the animator PARSFILEs. The keywords are also mentioned in the documentation for the animator library screens contained in Chapter 9.

Units

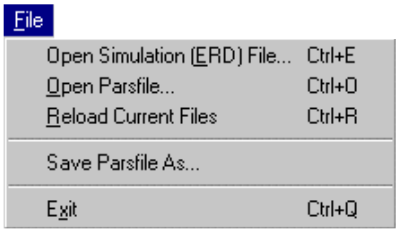
The animator requires all angles to be in degrees, and all coordinates to be in the same units of length. CarSimEd uses meters for animation length units. The animator allows user-defined scale factors for all coordinates, variables, and groups of shapes in a reference frame. This means that a mixture of units can be accommodated. For example, if some coordinates have units of meters and others millimeters, the values in millimeters can be associated with a scale factor of 0.001 to convert them to meters.

See the sections **Animator Reference Frames** and **Animator Shapes** in Chapter 9 for details on how to set scale factors.

File Menu

The file menu is used to read files, write files, and exit the program. Printing is not functional in the current version. To print a graphic,

1. copy it to the clipboard using the **Edit** menu item **Image Copy** (Ctrl+C)
2. paste it into another application (e.g., WordPad), and
3. print from the other application.



Open Simulation (ERD) File...

Select this command to open an ERD file with the Open dialog box. Recall that the six variables needed to define each reference frame are obtained from the ERD file produced by a solver program. If you read a new ERD file, the animator will show motions from that simulation without changing the camera point of view, shape descriptions, etc. The keyboard command is Ctrl+E.

Open Parsfile...

Select this command to open a PARSFILE (extension = PAR) with the Windows file browse dialog box. If you read a new PARSFILE, the animator will show the same motions as before, using the new camera point of view, shape descriptions, etc. as defined in the new PARSFILE. The keyboard command is Ctrl+O.

Reload Current Files

Use this command to restore the original settings by reloading the current ERD file and PARSFILEs. The keyboard command is Ctrl+R.

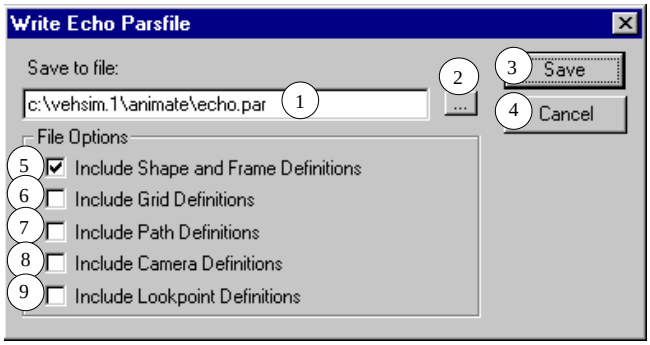
Besides restoring settings, this menu command can speed up the process of creating new shapes or other animator inputs, as described later in this chapter in the section **Testing Animator Data Sets** on page 77.

Save Parsfile As...

Select this command to save most of the animator settings. The animator data from the CarSimEd SGUI are spread over the animator libraries described in Chapter 9. Multiple shapes and reference frames are commonly used, which means that the animator program might read 20 or more PARSFILEs when it runs. This menu command creates a single PARSFILE with all of the information consolidated.

This PARSFILE can be used later as an input to the animator, to exactly recreate the current settings. The file can also be viewed in a text editor to debug inputs that cause problems, or to obtain camera and look-point coordinates that were set interactively using the **Coordinates** menu.

When you select this command, it displays the following dialog box. The numbered items in the dialog box are described below.



(1) Pathname for the file to be created. The default is `echo.par`, in the same folder as the animator program. You can type a new pathname in this field or use the browse button (2).

(2) Browse button. Click to bring up the Windows file browse dialog box to search your directory system for a suitable folder. If you select a folder using the browser, the resulting pathname is automatically entered into the file field (1).

(3) Save button. Click to create the Parsfile.

(4) Cancel button. Click to exit this dialog box without creating a file.

(5) Check box for shape and frame definitions. If checked when you click the Save button, all vehicle shape and frame data will be included in the file.

(6) Check box for grid definitions. If checked when you click the Save button, the current grid information will be included in the file.

(7) Check box for Path Definitions. This refers to the path input when a closed-loop driver model is used to steer the vehicle in a simulation. If checked when you click the Save button, the coordinates of the path will be included in the saved parsfile.

(8) Check box for Camera definitions. If checked when you click the Save button, the current camera reference frame and point coordinates will be included in the file.

(9) Check box for look-point definitions. If checked when you click the Save button, the current look-point reference frame and coordinates will be included in the file.

Exit

Exits the animator program. To use the keyboard, type the Windows-standard sequence: Alt+F, X or Ctrl+Q.

Edit Menu

The **Edit** menu supports the clipboard and sets preferences.

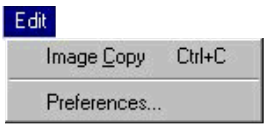
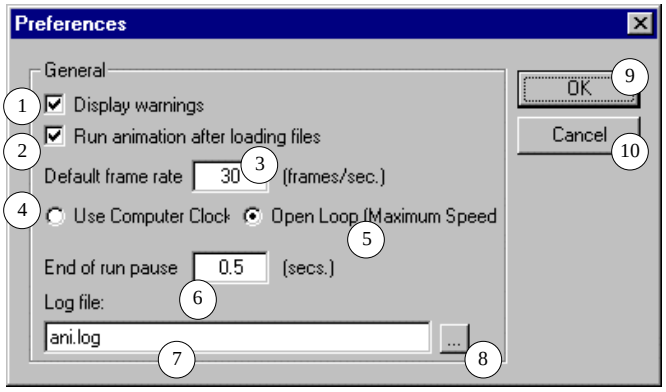


Image Copy

Select this command to copy the screen display to the clipboard as a bitmap. The keyboard command is Ctrl+C.

Preferences...

Select this command to set options for the animator that are saved when you quit the program. The preferences are stored in a file `Prefs.par`, described in Appendix E.



The numbered items in the dialog box are described below.

(1) Check box to display warnings. When checked, the animator will print messages to help diagnose errors in the input files.

(2) Check box to run after loading files. When checked, the animator starts running as soon as the files have been read. If the box is not checked, the animation must be started by typing Ctrl+S or using the **Animation** menu.

(3) Default frame rate. This value will be used when the animator is started. The frame rate can also be modified interactively (see Figure 6.3 on page 73). If the frame rate is larger than the rate defined by the time step in the ERD file generated during a simulation run, the animator will ignore this frame rate.

(4) Radio button for real time animation. If the animation needs to be slowed down to real time, select this button to slow it down to real-time. When this button is selected, the animator accesses the computer clock and waits, if necessary, to avoid running the animation faster than real time.

(5) Radio button for maximum speed animation. When this button is selected, the animator does not access the computer clock, allowing maximum display speed. Select this button if the animation is running slower than real time.

(6) End of run pause. The animator delays this amount of time at the end of a run before looping to start over. A pause of 0.5 sec or more gives a visual indication that the run has ended.

(7) Log file. Enter a name for an optional file written by the animator as it processes the inputs (use the adjacent button ... (8) to select a destination folder). The log file is sometimes helpful for debugging, but it is not needed for normal operation.

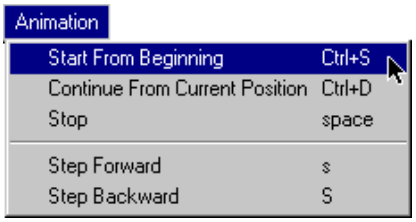
(8) Browse button. Click to bring up the Windows file dialog box to locate a destination folder for the log file (7).

(9) OK button. Click to close the dialog box and apply the new preferences.

(10) Cancel button. Click this to close the dialog box without changing anything (keep the previous preferences).

Animation Menu

The animation menu serves mainly to remind you of keyboard commands that control animation. Although the menu items are functional, you will usually find it quicker to use the indicated keyboard commands.



Start From Beginning

Select this command to restart the animation from the beginning of the simulation run. The keyboard command is Ctrl+S.

Continue From Current Position

Select this command to continue the animation if it has been stopped. The keyboard command is Ctrl+D.

Stop

Select this command to stop (pause) the animation. This is like being able to freeze time. While stopped, you can still use the **Coordinates** menu to move the camera around, zoom in and out, and change the look-point. The keyboard command is the space key.

Step Forward

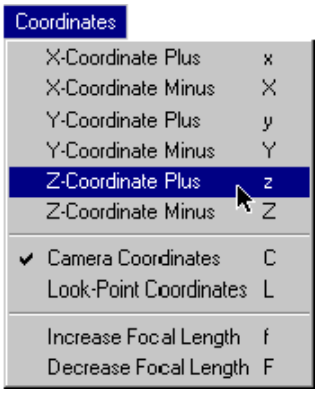
Select this command to step forward one increment in time, like a frame advance on a VCR. The keyboard command is the ‘s’ key.

Step Backward

Select this command to step backward one increment in time. The keyboard command is Shift+S.

Coordinates Menu

This menu serves mainly to remind you of keyboard commands that control the camera viewpoint. Although the menu items are functional, you will usually find it quicker to use the indicated keyboard commands.



The first six items are used to change the position of either the camera point or the look point, depending on which is checked further down in the menu. The currently affected point is checked in the menu, and is written in the animator status bar (at the bottom of the animator window).

The camera and look points are shown in Figure 6.1 on page 64. The camera point defines where the camera is located, and the look point defines the direction in which the camera is looking. For example, consider changing the Z coordinate (height) of either point. Increasing the Z coordinate of the camera has the effect of raising the camera, as if you are viewing the vehicle from a helicopter that rises. On the other hand, increasing the Z coordinate of the look point has the effect looking up. When you look up, everything in view shifts down.

X-Coordinate Plus

Select this command to increase the X coordinate of the camera or look point. The keyboard command is the ‘x’ key.

X-Coordinate Minus

Select this command to decrease the X coordinate of the camera or look point. The keyboard command is Shift+X.

> Note: The keyboard commands are assigned so plain key commands increase things, and using the shift key decreases things. For example, 's' steps forward, Shift+S steps backward; 'x' increases an X coordinate, Shift+X decreases it.

Y-Coordinate Plus

Select this command to increase the Y coordinate of the camera or look point. The keyboard command is the ‘y’ key.

Y-Coordinate Minus

Select this command to decrease the Y coordinate of the camera or look point. The keyboard command is Shift+Y.

Z-Coordinate Plus

Select this command to increase the Z coordinate of the camera or look point. The keyboard command is the ‘z’ key.

Z-Coordinate Minus

Select this command to decrease the Z coordinate of the camera or look point. The keyboard command is Shift+Z.

Camera Coordinates

Select this command to cause the camera position to be controlled by the first six menu items. The keyboard command is the Shift+C key.

Look-Point Coordinates

Select this command to cause the look-point position to be controlled by the first six menu items. The keyboard command is the Shift+L key.

Increase Focal Length

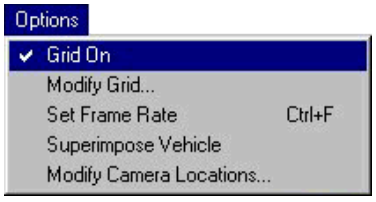
Select this command to increase the camera focal length. This is the same as zooming in. The keyboard command is the ‘f’ key.

Decrease Focal Length

Select this command to decrease the camera focal length. This is the same as zooming out. The keyboard command is Shift+F.

Options Menu

This menu has commands for setting several options in the animator operation.

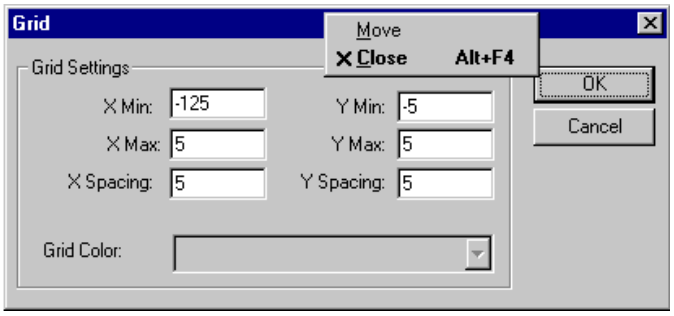


Grid On

The grid can be set to be displayed (on) or not displayed (off). Select this command to toggle the grid display.

Modify Grid...

Select this command to display a dialog box for setting properties of the grid.



Set Frame Rate

Select this command to display the following dialog box and control the rate at which the animator plays back a simulation run. The keyboard command is Ctrl+F. The items numbered in Figure 6.3 are described below.

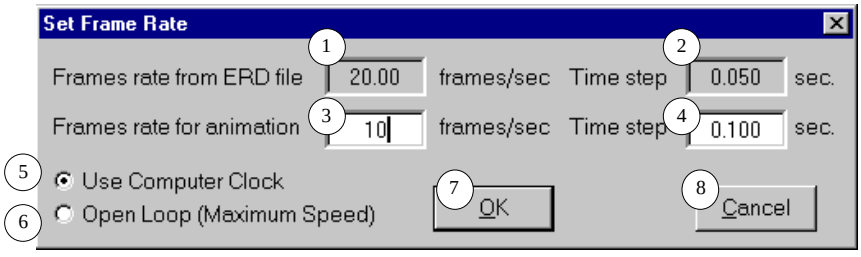


Figure 6.3. Setting the frame rate.

(1) Maximum frame rate possible, given the data in the ERD file. The frame rate is the reciprocal of the time step (2). This value is for reference only.

(2) Time interval between stored data points from the ERD file header. This value is specified in line 2 of the header of the ERD file. This is the minimum time possible between animation frames. This value is for reference only.

(3) Target frame rate for animation. If the animation is running too slowly, you can specify a **lower frame rate** to speed it up. Alternatively, you can use a higher frame rate to slow it down. Typically, frame rates of 10 to 30 frames per second can run in real-time on Pentium computers. If the target frame rate is very high (more than 100), the computer will not be able to refresh the screen in real time and the animation will run in slow motion. If you change this value, the animation time step (4) is automatically changed (the frame rate is the reciprocal of the time step).

The default frame rate is specified under the **Preferences** option in the **Edit** menu (see page 69).

(4) Animation time step. If the animation is running too slowly, you can specify a **larger time step** to speed it up. Use a smaller time step to slow it down. If you change this value, the animation frame rate (3) is automatically changed (the frame rate is the reciprocal of the time step).

> Note: If you specify a time step that is smaller than the value in the file (2), the value in the file is used. To run with the minimum time step (show every frame possible), you can enter a value of 0 in (4) and the animator will replace it with the value from the ERD file.

(5) Radio button for real time animation. If the animation is running too quickly, select this button to slow it down to real-time. When this button is selected, the animator accesses the computer clock and waits, if necessary, to avoid running the animation faster than real time.

> Notes: The process of checking the computer clock introduces a small delay. The delay depends on the computer and how your system is set up.
>
> If the animation is already running slower than real-time, checking the computer clock will slow it down even more. If the animation is running slower than real time and you want to speed it up, you must decrease the frame rate (3) or increase the time step (4). Also, select button (6) to eliminate the delay caused by checking the clock.

(6) Radio button for maximum speed animation. When selected, the animator does not access the computer clock, allowing maximum display speed.

(7) OK button. Click to exit the dialog box and apply the new settings.

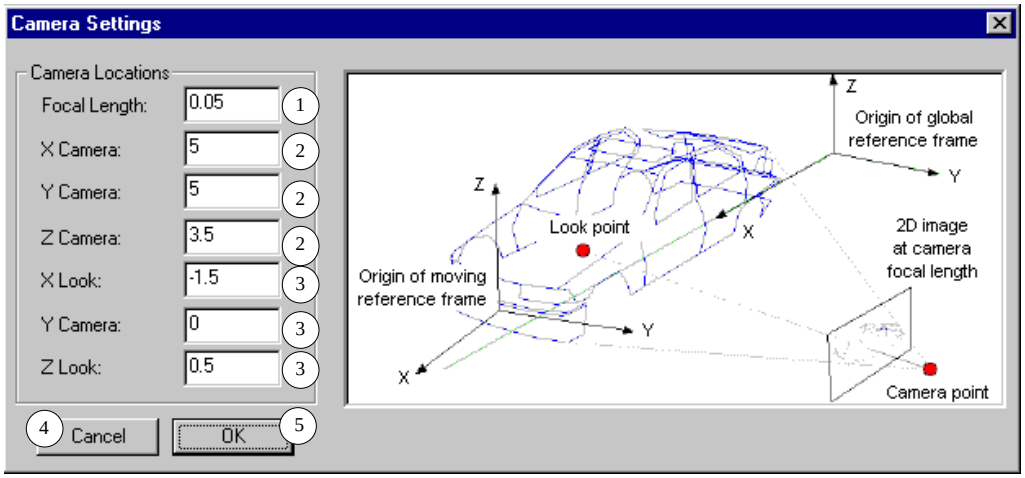
(8) Cancel button. Click to exit the dialog box without changing the settings.

Superimpose Vehicle

Select this command to toggle an option to draw new images without erasing old ones. This mode is useful with some configurations to show trajectories of vehicle motion, or to show amplitudes of oscillations.

Modify Camera Locations...

Select this command to display the following dialog box. The numbered items are described below.



> Notes: These settings are similar to those contained in the **Animator Camera Setup** library, described in Chapter 9.
>
> Units of length should be compatible with the coordinates used for shapes and reference as described in the section **Units** on page 66.

(1) Focal length. This is the distance from the camera point to the 2D plane used to project the 3D objects. See Figure 6.1 for a description of the geometry.

(2) X, Y, and Z coordinates of the camera point in its reference frame.

(3) X, Y, and Z coordinates of the look point in its reference frame.

(4) Cancel button. Click to exit the dialog box without changing the camera settings.

(5) OK button. Click to exit the dialog box and apply the new settings.

Help Menu

This menu is standard in Windows for providing information about a program.

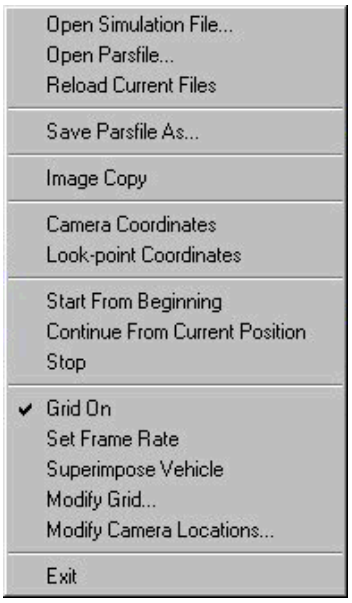
About Engineering Animator...

Select this option to display the current animator version and creation date.

Pop-Up Menu

Click on the right mouse button to display the following menu.

All items on this menu are copies of items on the main menus and perform the same functions when applied from the right-click pop-up menu. The menu is provided so you don't have to move the mouse as much to perform common actions.



Time Control Slider

Under the animator window there is a time-control slider that provides random access to the animation. Use it to jump to any point in a run. This can be convenient when working with long simulation runs, because you can go right to an event of interest without viewing the early parts.

(1) Start time from the input file.

② Finish time from the input file.

③ Current animation time. This is the simulation time associated with the currently displayed animation view.

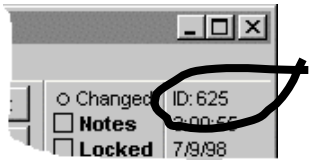
④ Slider control. The slider moves as the animation proceeds. Its relative position indicates graphically the time associated with the currently displayed view. You can use the left-button of the mouse to move the slider and change the current animation time.

Testing Animator Data Sets

The animation setup information is contained in the CarSimEd database but is displayed by the animator program. It is helpful to know how to use both the database and the animator to test and debug new animator settings.

Viewing the PARSFILE Tree

As shown in Figure 6.2 on page 66, all information about shapes, camera position, reference frames, etc. comes into the animator through PARSFILEs (CarSimEd-generated files with extension PAR). At the top level, the animator receives the name of an ERD file and the name of a PAR file. Both have the same base name—the number appearing in the upper-right corner of the **Runs** screen, as shown below.



The top-level PAR file is associated with the **Runs** screen and includes references to other PAR files. In fact, every blue link on the screen is represented in the PAR file with the keyword PARSFILE followed by a full pathname to another PAR text file. If any of those pathnames are not valid, the animator will generate an error message and quit without showing an animation.

Bad pathnames can be quickly located by using the PARSTREE program in CarSimEd. To run it, just click on the button with the tree icon: . Alternatively, go to the **Tools** menu and select the item **View Parstree**. The PARSTREE program brings up a window with two panes, similar in format to the Windows Explorer.

For the example shown in Figure 6.4, the file 211.par is marked to indicate an invalid reference. The panel on the right displays the contents of the selected PAR file, 212.par. From reading the PAR file that references the nonexistent file, it is seen to be in the animator FRAMES folder. Inspection of the FRAMES library should reveal that one of the data sets is nonexistent, causing the bad link.

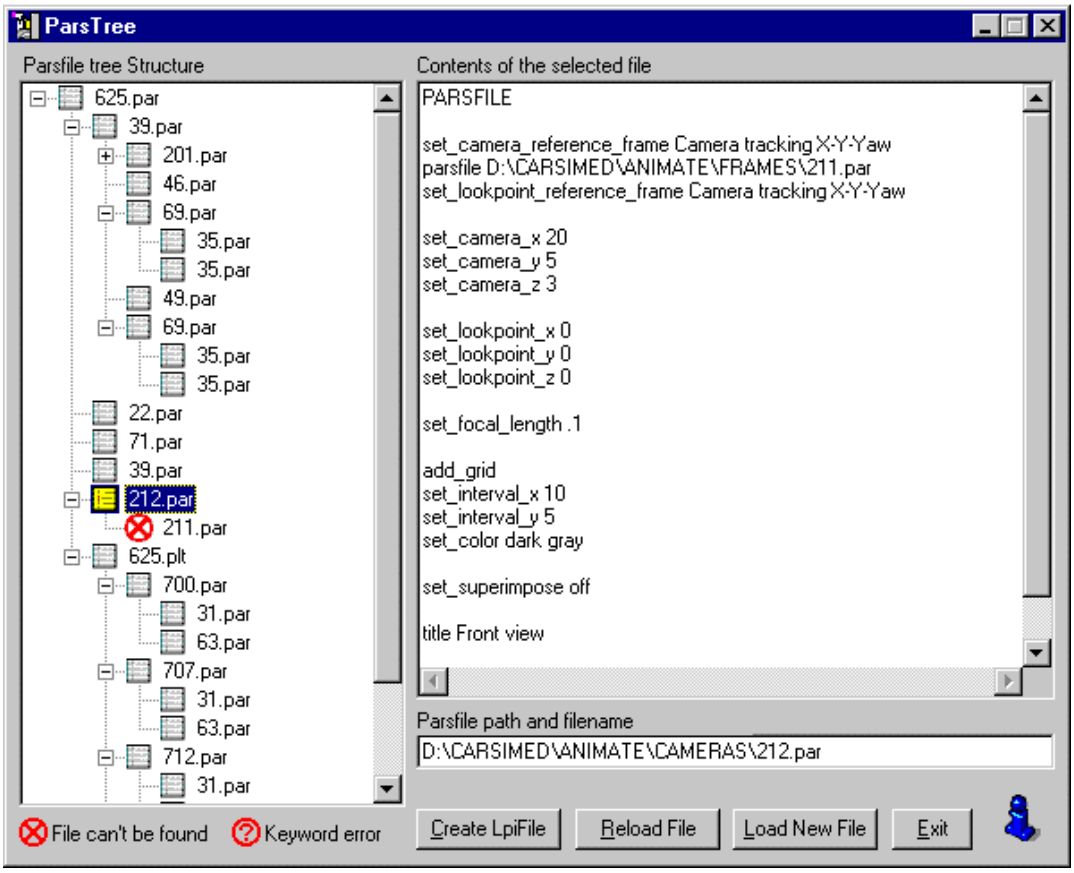


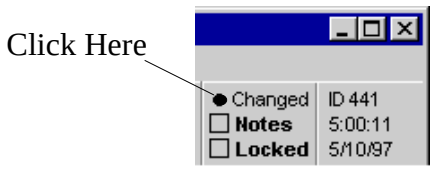
Figure 6.4. Parstree window.

Toggling Between the Database and the Animator

The basic method for editing shapes and testing animator data sets is as follows:

1. Create animator settings using the animator libraries described in Chapter 9.
2. Link to the new settings on the **Cars** screen (via the **Animator Group** link). To test new camera settings, link to the new settings from the **Runs** screen (via the **Camera Setup** link).
3. Go to the **Runs** screen and click the **Animate** button. The run should use the Car data set with the link to the new animator settings.
4. View the animation. When you are ready to modify the settings, stop the animation. Press the space key or use the **Stop** command from the **Animation** menu.
5. Without quitting the animator, switch back to the CarSimEd **Runs** screen. (Use Alt+Tab in Windows to switch between running programs.)
6. Within CarSimEd, navigate to the screen with the animator data to be edited (e.g., an animator shape data set).

7. Edit the data in the CarSimEd data set. As soon as you modify any data and move the cursor to another field, you will see a red circle in the upper-right corner of the window, as shown below.



8. Click on the red circle in the upper-right corner. This causes CarSimEd to update the PARSFILE using the new data from the screen.

> Note: Normally, communication between the database and the animator is transparent. All necessary files are properly updated when you click the **Animate** button in the **Runs** screen. Here, you are bypassing the normal operation, and must force the SGUI to update the PARSFILE before it would normally do so.

9. Switch back to the animator (use Alt+Tab).

10. Select the menu item **Reload Current Files** under the **File** menu (or type Ctrl+R) to see the effect of the new data.

11. Optionally switch back to the SGUI screen and repeat steps 6 through 9 as needed.

7. The Plotter

You can view plots of the CarSimEd simulation results with a program called Windows Engineering Plotter (WinEP). WinEP is a versatile tool you can use to plot any two variables against each other. You can also overlay data from the same file or different files.

Updates

The plotting program is used in a number of software packages. The latest version is generally available from the Internet at:

http://www.trucksim.com/winep/index.html

Updates of the plotter program are "backward compatible." If you download a newer version, replace the file `Winep.exe` in the `Plot` folder in CarSimEd.

Overview of Operation

WinEP has a workspace defined by a main window with a menu bar. The main window contains plot windows, each showing plots made with a single set of X-Y axes. Figure 7.1 shows the main WinEP window with four example plot windows.

Each plot window contains a graphical representation of one or more X-Y data sets. An X-Y data set is a series of X and Y values obtained from a data file. The X-Y data sets can come from the same file or from different files. The X values in each data do not have to be the same, and the data sets do not have to contain the same number of points.

WinEP reads the data from ERD and text files and has completely automated scaling, formatting, and labeling capabilities. However, labeling options are limited for non-ERD text files.

> Note: If a file is not in ERD format, it can be converted by adding several lines of text to the beginning of the file. A description of the ERD file format is provided in Appendix C.

Once a plot window is created, the X-Y data sets can be transformed by subtracting offsets or by applying high-pass or low-pass filters. However, the data sets cannot be deleted or replaced. If you create a plot with different data sets, the new plot will have a new window.

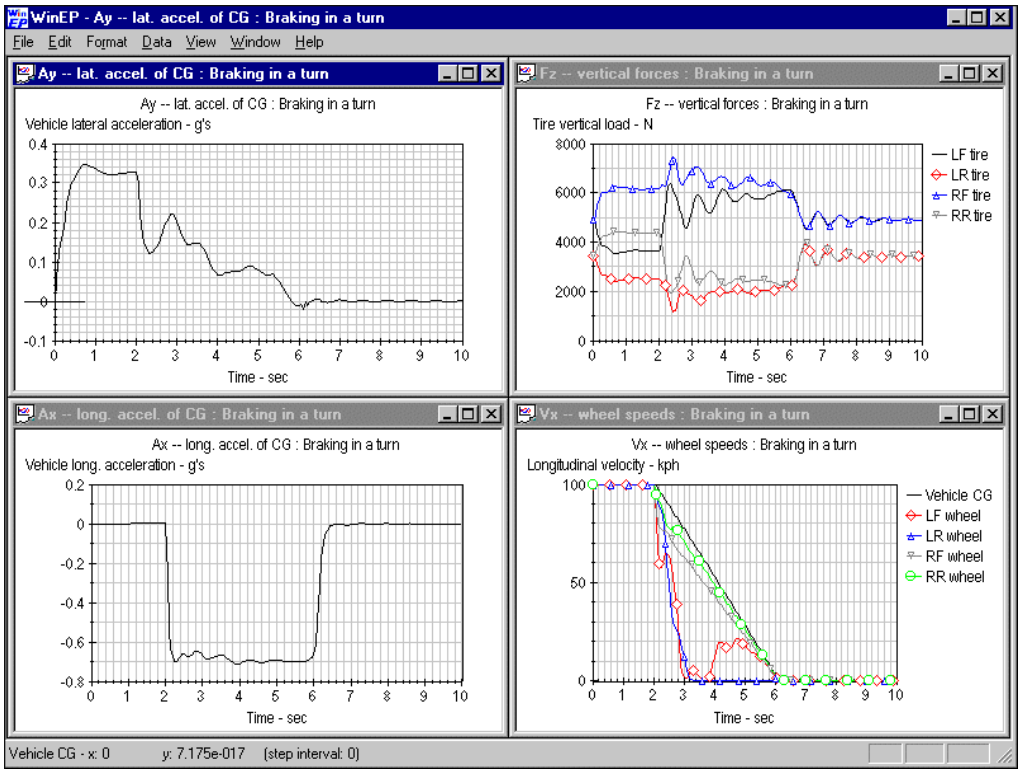

WinEP - Ay -- lat. accel. of CG : Braking in a turn
File Edit Format Data View Window Help
Ay -- lat. accel. of CG : Braking in a turn
Ay -- lat. accel. of CG : Braking in a turn
Vehicle lateral acceleration - g's
Time - sec
Fz -- vertical forces : Braking in a turn
Fz -- vertical forces : Braking in a turn
Tire vertical load - N
LF tire
LR tire
RF tire
RR tire
Time - sec
Ax -- long. accel. of CG : Braking in a turn
Ax -- long. accel. of CG : Braking in a turn
Vehicle long. acceleration - g's
Time - sec
Vx -- wheel speeds : Braking in a turn
Vx -- wheel speeds : Braking in a turn
Longitudinal velocity - kph
Vehicle CG
LF wheel
LR wheel
RF wheel
RR wheel
Time - sec
Vehicle CG - x: 0 y: 7.175e-017 (step interval: 0)

Figure 7.1. The WinEP workspace.

In addition to the X-Y data, each plot window has a set of formatting attributes, such as the colors of the lines, the size of the labeling text, the type of grid, etc. You can show the data points with connecting lines, symbols, or both. You can use linear or log scaling and you can choose from several axis types. The formatting can be set independently for each plot, and changed interactively at any time.

You can print the WinEP plots or copy them to the clipboard.

There are no built-in limits to the number of active plot windows that can be maintained by WinEP. The only limits are imposed by computer memory.

Batch and Interactive Operation

WinEP can be started in one of two modes. When you click the **Plot** button from the CarSimEd **Runs** screen or the **Make Plots** button from the **Plot Setup: Batch** screen, CarSimEd starts WinEP in batch mode. It sends WinEP a file with information that you specified on the **Runs** screen. Based on the contents of this file, WinEP initially creates one or more plots. Once the initial batch of plots is made, WinEP reverts to interactive mode. In the interactive mode you can view the plots, print them, and change their formatting. You can also create new plots.

You can also run WinEP as a normal Windows application. You start in this mode by double-clicking on the file `Winep.exe`. When started in this way, the WinEP

workspace is empty and you must use a menu command to define the data for the first plot. A better way to start is by double-clicking on a file with the extension ERD. This starts the WinEP program and automatically loads the selected ERD file. You must then pick X and Y variables from the file to create the first plot.

> Note: The file `Winep.exe` must be associated with the file type ERD if you want to launch it by double-clicking on an ERD file. The association is made automatically as part of the normal CarSimEd installation. You would need to re-establish the association only if you move the CarSimEd directory, or if you copy the WinEP program to another computer that does not have CarSimEd on it.

Appendix D describes the batch control file for WinEP, which is created automatically by CarSimEd, and the keywords that are recognized by WinEP.

The rest of this chapter covers the interactive use of WinEP.

Zooming

WinEP usually creates each plot window with the axes scaled to show all data points. To zoom in, use the mouse to create a zoom rectangle. Press the left button to anchor one corner at the current cursor location, then drag with the button still pressed to define the rectangle. When the rectangle covers the desired range, release the button. WinEP will re-draw the plot using the region of the rectangle as the new limits for the axes.

Figure 7.2 shows the appearance of a plot with a zoom rectangle drawn, just before the mouse button is released. Figure 7.3 shows the same plot window immediately after the button is released.

> Note: WinEP rounds off the maximum and minimum values for the axes. The new plot is drawn to include the full range of the rectangle you drew with the mouse, plus some additional range to round off the axes. For example, the maximum Y value from the zoom rectangle shown in Figure 7.2 is about 4500. The maximum value in the zoomed plot of Figure 7.3 is rounded up to 5000.

If you click in a plot window without drawing a rectangle, WinEP re-draws the screen without changing the scale.

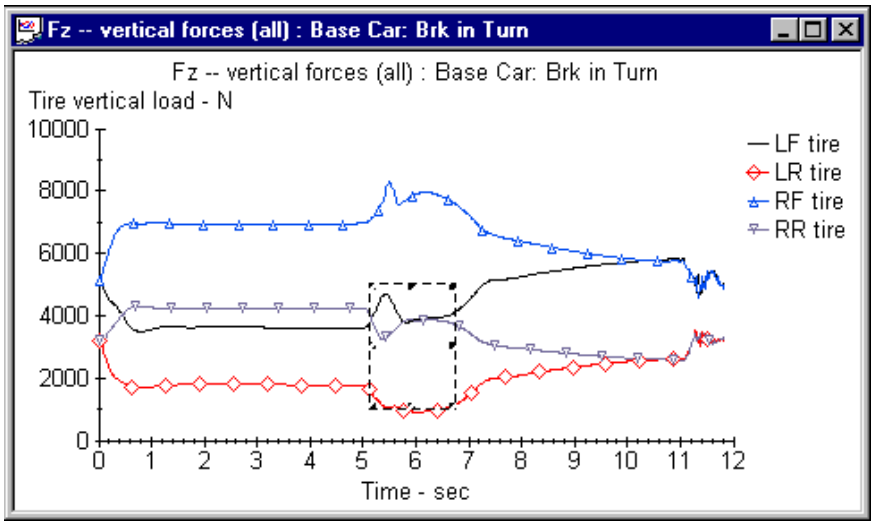


Figure 7.2. Plot before zooming.

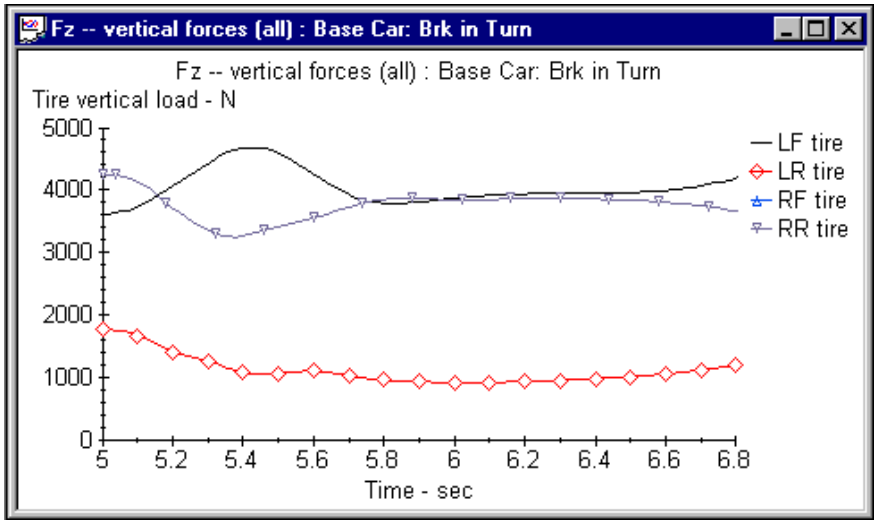


Figure 7.3. Plot after zooming.

Tool Bar

WinEP includes a toolbar with buttons for applying some of the most commonly used commands. Place the cursor over a button to read its function in a pop-up box and also in the status bar.

All toolbar buttons provide actions that duplicate menu items. The actions are described in the following descriptions of the menu items. When a menu item has an associated toolbar button, the button is shown in the heading for the menu item. If the toolbar does not appear in your plot window, click on the **View** menu command **Toolbar**.

Printing Plots

Making a Hard Copy

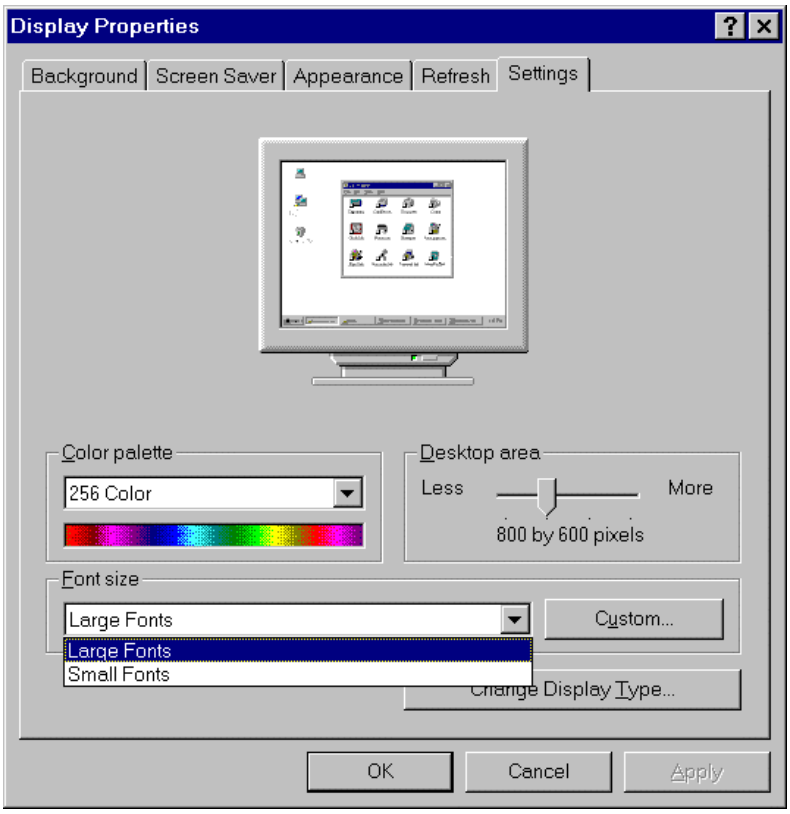
1. If there are more than one plot windows, select the plot window of interest.
2. Select the menu item **Print** from the **File** menu. The keyboard command is Ctrl+P. The plot in the current window will be sent to the current Windows printer.
3. To print additional plots, repeat steps 1 and 2 for each plot window of interest.

Controlling the Printed Plot Size

Plots in WinEP are scaled on the screen to fit in their windows, which you can size within the limits of your monitor resolution. To determine how a plot will be printed, select the menu item **Print Preview** under the **File** menu.

When printed, the size of the plot on paper depends on a number of factors.

- The window dimensions on the screen (height and width). You adjust the window size as with any other window, using the Maximize box in the upper-right corner of the window and by dragging borders with the mouse.
- The Windows display settings. To bring up this dialog box:
    1. Click in the Windows **Start** button to display the **Start** menu.
    2. Click on the **Settings** menu item to display a submenu.
    3. Select the submenu item **Control Panel** to display a folder of control panels.
    4. Select the control panel **Display**. When the Display settings window appears, click on the **Settings** tab.



Note: In general the Small Fonts setting is recommended for the CarSimEd software. Problems sometimes occur with the display of CarSimEd screens with some combinations of hardware and monitor settings when Large Fonts are selected. Also, it is more complicated to estimate the size of printed plots.

- The paper size and orientation (portrait or landscape). To specify the paper size and orientation, select the **Print Setup** menu item from the **File** menu in WinEP. This brings up a dialog box with settings specific to the printer.
- The percent reduction, which can be specified on some printers (particularly PostScript printers). Print reduction, if available, is found from the dialog box obtained by selecting **Print Setup** from the **File** menu.

File Menu

The file menu is used to read files, write files, print windows, and exit the program.

Load ERD File...

Select this command to open an ERD file with the Windows file browser dialog box. After the file is opened, a dialog box appears that you use for selecting variables from the file to plot. The keyboard command is Ctrl+E.

Load Batch File...

Select this command to process a batch plot file with the Windows file browse dialog box. When created automatically by CarSimEd, batch plot files have the extension PLT and are located in the directory containing the library where the **Plot** button was clicked (either `Runs` or `Batch\Plot_bat`). When loaded, the batch file will cause WinEP to generate one or more plots. The keyboard command is Ctrl+B.

Load Plot Format...

Formatting information for plots can be stored in files. These files can be created from within WinEP using the **Save Plot Format** command (from the **File** menu) and also from within the CarSimEd database (see the section **Plot Format** in Chapter 9).

Select the **Load Plot Format** command to load one of these format files. This command brings up the Windows file browser dialog box to open a file with the extension PAR. When loaded, WinEP will redraw the current plot with whatever new formatting information was loaded. The keyboard command is Ctrl+O.

> Note: Format files created within the CarSimEd graphical user interface are in the CarSimEd folder `Plot\Format`.

Load Transform Settings

Transform and offset information for plots can be stored in files. These files can be created from within WinEP using the **Save Transform Settings** command (from the **File** menu).

Select the **Load Transform Settings** command to load one of these settings files. This command brings up the Windows file browser dialog box to open a file with the extension PAR (default file is PlotTran.par). When loaded, WinEP will redraw the current plot with whatever new settings information was loaded. The keyboard command is Ctrl+T.

Close Top Window

This closes the top (active) plot window. The keyboard command is Ctrl+W.

Save Plot Format…

Use this command to save the current plot formatting information. The default extension is PAR. The formatting can be applied in the future using the **Load Plot Format** command from the **File** menu. When loaded, the formatting file will cause WinEP to redraw the current plot with the new format. The keyboard command is Ctrl+S.

Save Plot Data…

Use this command to save the current X-Y data set in a text file. It is a handy way to extract data from an ERD file, two channels at a time. A similar option, **show numbers**, is available under the **View** menu for viewing the data.

Save Channel List…

This command creates a text file listing all of the channels in an opened ERD file. WinEP brings up a dialog box for saving a text file with the list of channels. A similar option, **Show Numbers**, is available under the **View** menu for viewing the data.

Save Transform Settings

Use this command to save the current data transforms and offset settings. The default extension is PAR (the default file is `PlotTran.par`). The settings can be applied in the future using the **File** menu command to **Load Transform Settings**. When loaded, the formatting file will cause WinEP to redraw the current plot with the new settings.

Save All Images

Use this command to save all the plots as image files with BMP format. The folder used for saving images can be set using the **Edit** menu command **Preferences**. The keyboard command is Ctrl+I.

Print...

Use this command to make a hard-copy of the active plot using the current Windows printer. The keyboard command is Ctrl+P.

Print Preview

This command opens a window showing how the active plot would be printed using the current print settings.

Print Setup...

This command brings up the Windows dialog box for changing printer information.

Exit

Use this command to exit WinEP. To use the keyboard, type the keyboard command Ctrl+Q, or Alt+F4, or Alt+F, X.

> Note: Alternatively, you can click the in the upper right hand corner of this screen to return to the **Runs** screen.

Edit Menu

The **Edit** menu has commands to support the clipboard and to set preferences. When a dialog box is in front, the **Edit** menu cannot be used with the mouse. However, the keyboard commands are functional and can be used to cut, copy, and paste within text fields.

Undo

Undoes the most recent change in a dialog box. The keyboard command is Ctrl+Z.

Cut

Cuts currently selected text to the clipboard. This item is only in effect when editing text in dialog boxes. Modification of the graphic display is not supported. The keyboard command is Ctrl+X.

Copy

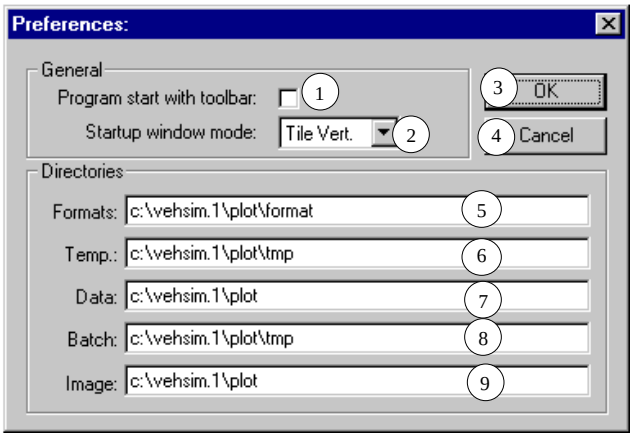
This command copies the plot from the active plot window to the clipboard. The graphic is copied as a bitmap. If a dialog box is active, this command copies currently selected text to the clipboard. The keyboard command is Ctrl+C.

Paste

Pastes text from the clipboard into the current cursor location. If text is selected, it is replaced by the contents of the clipboard. This item is only in effect when editing text in dialog boxes. Modification of the graphic display is not supported. The keyboard command is Ctrl+V.

Preferences

This command brings up the **Preferences** dialog box. When you quit WinEP, the settings from this dialog box are automatically saved to a file named `Epprefs.txt`, which stores these settings.



The numbered items in the dialog box are described below.

(1) Toolbar display mode. When checked, the toolbar is displayed when the program starts. Otherwise the toolbar is hidden and you must select the **Toolbar** item on the **View** menu to show it.

(2) Startup window mode. You use this menu to specify the initial appearance of the plot windows when WinEP is started in batch mode. For the Cascade setting, the individual plot windows are smaller than the WinEP workspace and overlap each other. For the Maximize setting, the active plot window is maximized to fill the WinEP workspace,

hiding any other plot windows. For the Tile settings, multiple plots are tiled to fill the WinEP workspace.

③ **OK** button. Click to close the dialog box and apply the new preferences.

④ **Cancel** button. Click to close the dialog box without changing anything (keep the previous preferences).

⑤ Folder from where the last format file was read. The next time the **File** menu command **Load Plot Format** is used, WinEP will open the file browser starting with this folder.

⑥ Folder where WinEP creates temporary files.

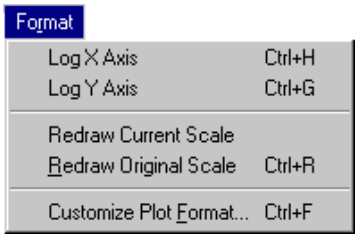
⑦ Folder from where the last ERD file (with numerical data to plot) was read. The next time the **File** menu command **Load ERD File** is used, WinEP will open the file browser starting with this folder.

⑧ Folder from where the last batch file was read. The next time the **File** menu command **Load Batch File** is used, WinEP will open the file browser starting with this folder.

⑨ Folder from where image files will be written by the **File** menu command **Save All Images** (CTRL+I).

Format Menu

This menu has commands for controlling the plot appearance.



Log X Axis

Use this command to toggle between linear and log scaling for the X axis of the active plot. If the log axis is selected, the menu item has a check mark. The keyboard command is Ctrl+H.

> Note: WinEP will ignore the X axis log option and revert to the linear option if there are one or more zeroes or negative X values in the data set. However, if you zoom in to exclude the zero and negative values, the log option can be used for the data shown.

Log Y Axis

Use this command to toggle between linear and log scaling for the Y axis of the active plot. If the log axis is selected, the menu item has a check mark. The keyboard command is Ctrl+G.

> Note: WinEP will ignore the Y axis log option and revert to the linear option if there are one or more zeroes or negative Y values in the data set. However, if you zoom in to exclude the zero and negative values, the log option can be used for the data shown.

Re-Draw Current Scale

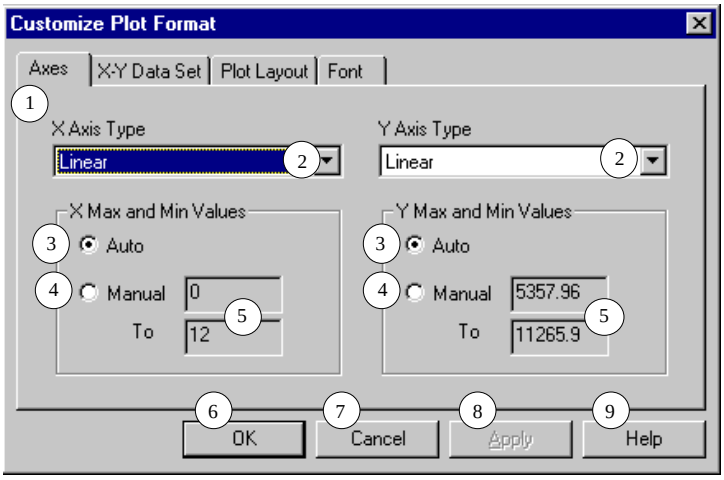
Use this command to re-draw the active plot with the current formatting. It is sometimes needed if the window display gets corrupted by another window being in front. You can accomplish the same thing by clicking in the plot area with the mouse.

Re-Draw Original Scale

Use this command to re-draw the active plot without any zooming. The current scaling option is used, as specified with the **Customize Plot Format** option under this menu. The keyboard command is Ctrl+R.

Customize Plot Format…

Use this command to display a preferences dialog box for setting most plot formatting options. The keyboard command is Ctrl+F. This dialog box is used for four sets of data, selected with tabs.



(1) Click this tab to toggle the axes scaling and to type in the desired range of axes values in the next plot.

(2) X, Y axis type. These controls define whether the axis will be drawn with a linear or log scale. They duplicate the **Format** menu commands **Log X Axis** and **Log Y Axis**.

(3) Autoscaling (X and Y axes). Check these buttons to cause WinEP to scan the data and size the axis to include the full range of data. This setting is recommended for routine use.

(4) Manual scaling (X and Y axes). Check these buttons to scale an axis according to the range specified in the adjacent fields (5). This setting is provided for situations in which you wish to make plots with the same axes, regardless of the range covered by the data.

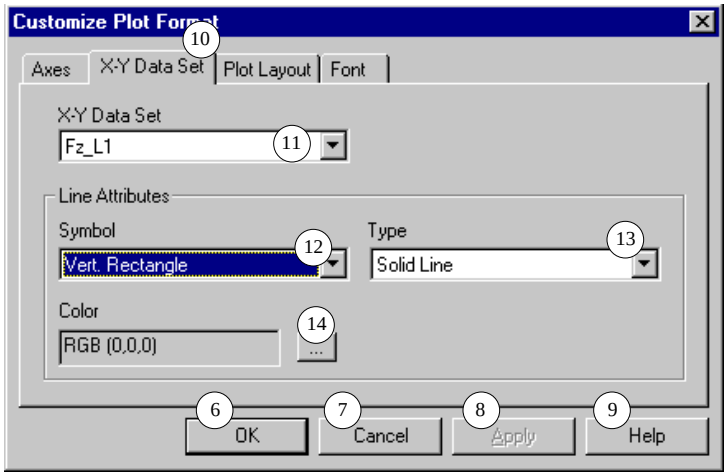
(5) Range of data values covered by an axis. Specify minimum and maximum values for scaling the axes. These are used only when the manual scaling button (4) is selected.

(6) **OK** button. Click to close this dialog box and apply the settings as displayed.

(7) **Cancel** button. Click to close this dialog box and revert to the previous settings.

(8) The **Apply** button is not functional in the current release.

(9) The **Help** button is not functional in the current release.



(10) X-Y Data Set tab. Click this tab to show the settings that control the display of a data set.

> Note: An X-Y Data set is a set of X and Y values from a specific file. Each data set is represented graphically in a plot. When multiple data sets are shown in the same plot, they can be distinguished by assigning them different display attributes.

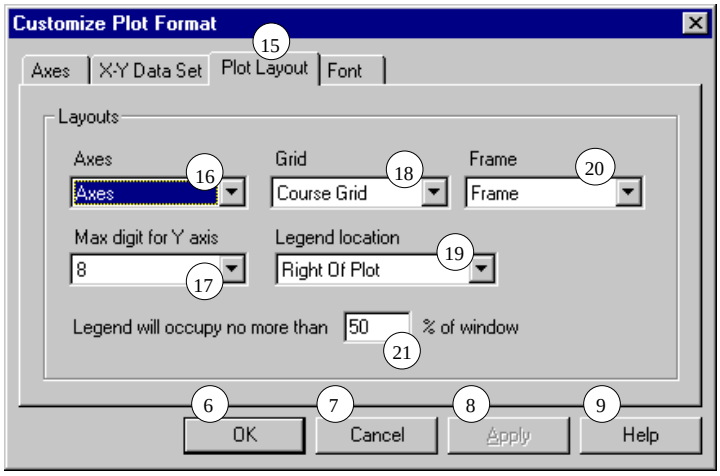
(11) X-Y Data set. Press the button ▼ to display a pull-down menu with the list of Y variables for each data set. Select the data set to display and optionally edit its attributes.

(12) Symbol. This field shows the symbol currently associated with the data set (11). Press the button ▼ to display a pull-down menu of available symbols. Choose one to change the current symbol.

(13) Line type. This field shows the type of line currently associated with the data set (11). Press the button ▼ to display a pull-down menu of supported types, and choose one to change the way connecting lines are drawn between the data points.

(14) Color. This field shows the color currently associated with the data set (11), in terms of RGB values (24-bit color, composed of components of Red, Green, and Blue). Press the button ... to display a palette of colors. Then pick a color to change the display color of the line and/or symbols.

After selecting a color, click the **OK** button to return to the format dialog box. The RGB values for the selected color will be shown.



(15) Plot Layout tab. Click this tab to show the settings that control the display of the axes, frame, and grid.

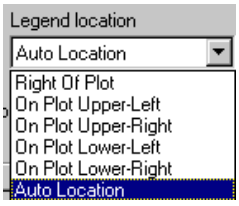
⑯ Axes. Press the button to display a menu with the choices: Axes or No Axes. The No Axes option doesn't work unless you also select No Grid with the Grid pull-down menu ⑱ and No Frame with the Frame pull-down menu ⑳.

⑰ Max digits for Y axis. This is the number of digits set aside for the tick labels of the Y axis. For example, if set to 3, WinEP draws the Y axis of every plot with enough space for 3-digit labels for the Y axis. If set to 0, WinEP determines the spacing automatically.

A value of 0 is recommended unless you want all plots to be drawn in exactly the same place relative to the left edge of the window.

⑱ Grid. Press the button to display a menu with the choices: No Grid, Coarse Grid (grid lines at major tick marks), or Fine Grid (grid lines at minor tick marks).

⑲ Legend location. Press the button to display a menu to choose among five possible locations for the legend used to identify the X-Y data sets in overlay plots. The sixth option is to let WinEp select the location automatically.



The legend and other labels in the plot are shown in Figure 7.4.

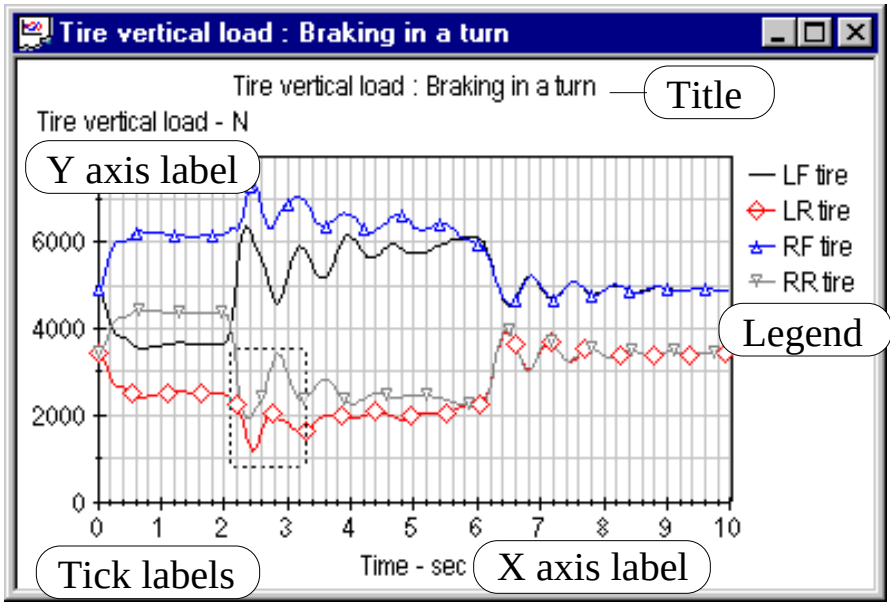
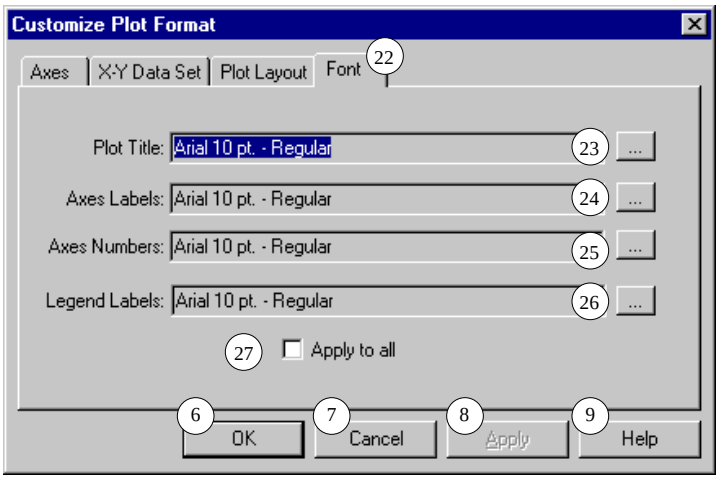


Figure 7.4. Names of plot labels.

(20) Frame. Press the button to display a menu to specify how the plot area is enclosed with a frame. The options are No Frame, Frame, and Frame with ticks. The No Frame option doesn't work unless you also select No Grid (18).

(21) Size limit for legend. When WinEP draws the legend for overlay plots to the right of the plot area, it uses whatever space is needed. If the labels are lengthy, or if the window is small, this can limit the horizontal space available for the plot. Use this field to set a maximum amount of window space that can be taken for the legend.

This setting does not affect the plots unless: (1) there are at least two X-Y data sets being overlaid, (2) the specified position (19) is "Right of Plot," and (3) the length of the longest label exceeds the specified percentage.



(22) Font tab. Click this tab to show the settings that control the text used to label the axes and data in the plot. The four types of labels, controlled with items (23) - (26), are shown in Figure 7.4.

(23) Plot title font attributes. Click the selector button ... to bring up a Windows font selection dialog box, as shown in Figure 7.5. The properties selected in the box will be applied to the plot title. After choosing the font properties, click the **OK** button to make the change, or click the **Cancel** button to keep the original properties.

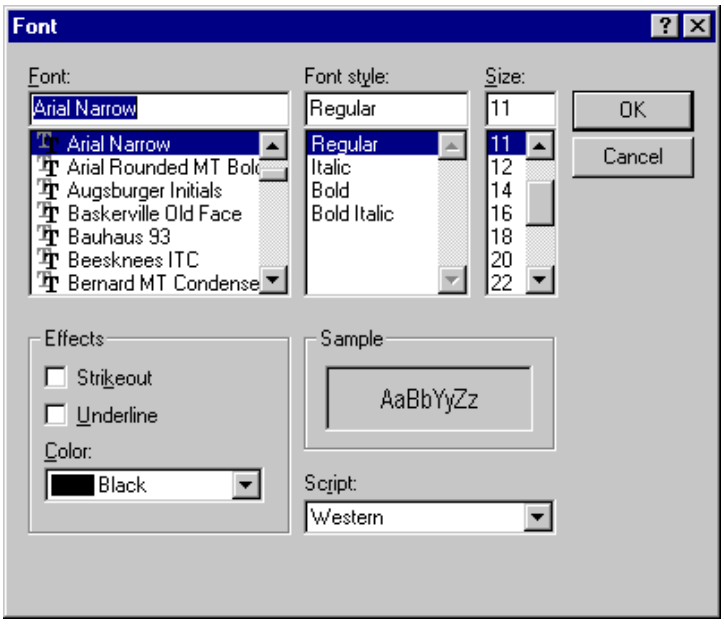


Figure 7.5. Font dialog box.

(24) Axes labels. This is similar to the plot title button (23), except that the settings made using the Windows font dialog box (see Figure 7.5) are applied to the text used to label the X and Y axes.

(25) Tick labels. This is similar to the plot title button (23), except that the settings made using the Windows font dialog box (see Figure 7.5) are applied to the text associated with the tick marks.

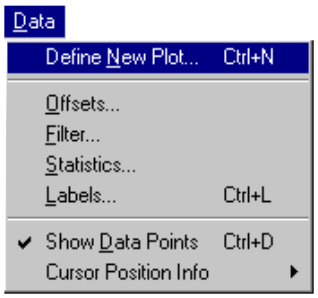
(26) Legend labels. This is similar to the plot title button (23), except that the settings made using the Windows font dialog box (see Figure 7.5) are applied to the text used to label the individual data sets in the legend.

(27) Apply to all check box. When this is checked, changing font properties using any of the buttons ... ((23) - (26)) will change all of the text in the plot.

> **Note:** To use this function, you must check this box before making the selection in the font dialog box shown in Figure 6.5.

Data Menu

The **Data** menu has commands that involve the numbers to be plotted and the associated labels from the data files. The commands involve the selection of X-Y data sets, possible transformation of the numbers, labels for the plot, and viewing the numerical values using a cursor.



Define New Plot...

Select this command to bring up the **Channel Select** dialog box. The keyboard command is Ctrl+N. Use this dialog box to create a list of X-Y data sets to plot. Each data set consists of three parts: (1) a Y channel, (2) an X channel, and (3) a file name. The numbered items in the dialog box are shown in Figure 7.6 and are described below.

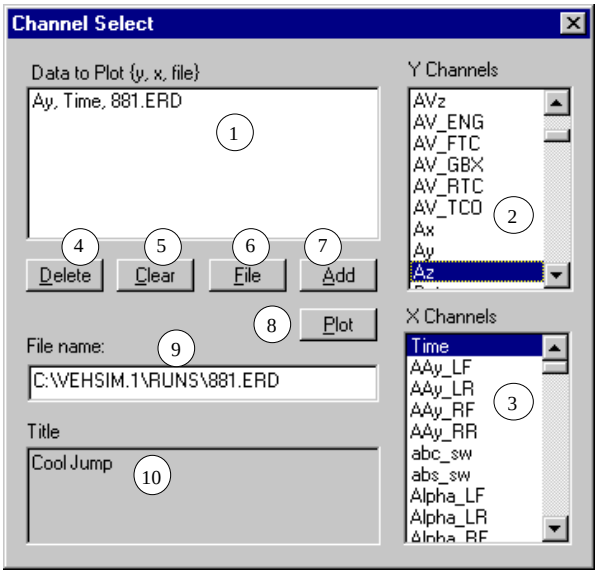


Figure 7.6. Channel Select dialog box.

(1) List of X-Y data sets to plot. This list shows the data sets that will be plotted. You cannot edit it directly, but can modify it using the buttons (2) - (7).

(2) List of variables in the current ERD available for defining the Y coordinate in a plot. To select a variable for plotting on the Y axis, click on its name. Double-clicking in this list has the same effect as clicking the **Add** button (7).

(3) List of variables in the current ERD available for defining the X coordinate in a plot. To select a variable for plotting on the X axis, click on its name. Double-clicking in this list has the same effect as clicking the Add button (7).

(4) **Delete** button. Click to delete a selected data set from the list of data to plot (1). This button has no effect unless a line is selected in the list (1).

(5) **Clear** button. Click to delete all data sets from the list of data to plot (1).

(6) **File** button. Click to bring up the Windows file browser dialog box to choose an ERD file or text file with plot data. This button has the same effect as going to the **File** menu to select the command **Load ERD File**.

> Note: While the dialog box is displayed, the WinEP menus cannot be used. Therefore, the File button is the only way to open a new file from within this dialog box.

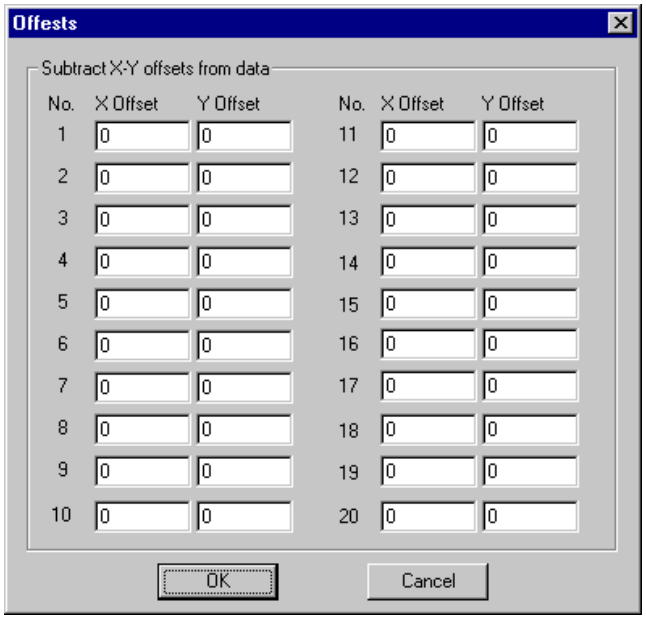
(7) **Add** button. Click to add the selected X-Y data sets to the list of data to plot (1). The new data set has the currently selected Y channel (2), the currently selected X channel (3), and comes from the currently open file (shown in (9)).

(8) **Plot** button. Click to close the dialog box and make a new plot using the current list of X-Y data sets (1). The current settings are not cleared when you make a plot—they remain in place if you bring up the **Channel Select** dialog box again.

(9) File name. This field shows the name of the current data file. The name can be copied to the clipboard, but there is no reason to edit it directly. To change the file, you must click the **File** button (6).

(10) Title. This field shows the title from the data file, if (1) the file is an ERD file, and (2) the optional TITLE keyword is used in the ERD header. All ERD files created by CarSimEd models show the title of the corresponding **Runs** data set.

Offsets...

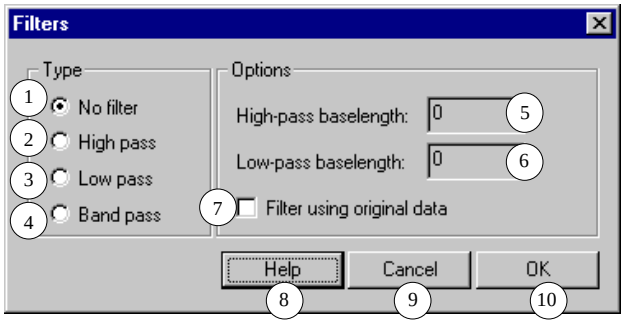

Offests
Subtract X-Y offsets from data
No. X Offset Y Offset No. X Offset Y Offset
1 0 0 11 0 0
2 0 0 12 0 0
3 0 0 13 0 0
4 0 0 14 0 0
5 0 0 15 0 0
6 0 0 16 0 0
7 0 0 17 0 0
8 0 0 18 0 0
9 0 0 19 0 0
10 0 0 20 0 0
OK
Cancel

Select this command to bring up the Offsets dialog box. Use this dialog box to specify offsets for up to 20 data sets laterally (X) or vertically (Y). The values you enter in these fields will be subtracted from the values obtained from the data file. For any fields left blank, no offset is applied.

When you are finished specifying the offsets, click the OK button. These offsets will be applied to data in the active plot.

Filters...

Select this command to bring up the **Filters** dialog box. Use this dialog box to filter the Y values in the data sets of the active plot using a moving average. The numbered items in the dialog box are described below.



(1) **No filter** button. When this button is selected the data are not modified.

(2) **High pass** button. When this button is selected, the Y variables are processed with a high-pass filter. A high-pass moving average filter works by applying a low-pass smoothing filter (with baselength (5)), and then subtracting the smoothed points from the original, leaving deviations from the smoothed data. This removes low frequencies, bias, and drift. It leaves the high frequencies intact.

(3) **Low pass** button. When this button is selected, the Y variables are processed with a low-pass filter. A low-pass moving average filter transforms a series of numbers by replacing each number with the average value, where the average is taken over some number that corresponds to the specified baselength (6). The number of points used in the average is the baselength, divided by the interval between points. The averaging process smoothes the data, removing high frequencies and passing low frequencies.

(4) **Band pass** button. When this button is selected, the Y variables are processed with a band-pass filter. The data are processed by a high-pass filter using the high-pass baselength (5), and then by a low-pass filter using the low-pass baselength (6).

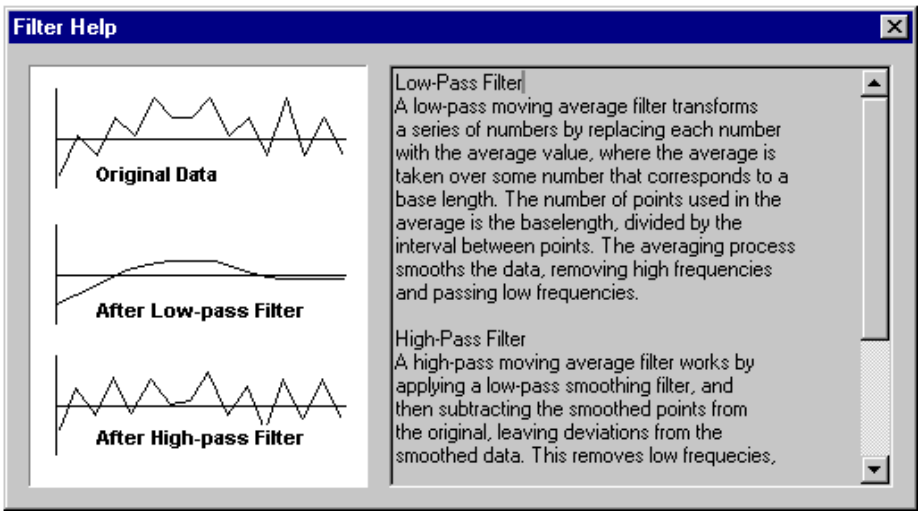
(5) High-pass baselength. This is a reference used to define the scope of a moving average. It has the same units as the variable plotted on the X axis. For example, if the X variable is

time with units of seconds, the baselength is the number of seconds covered by the moving average. This baselength is also used for band-pass filtering.

6. Low-pass baselength. This is a reference used to define the scope of a moving average. It has the same units as the variable plotted on the X axis. For example, if the X variable is time with units of seconds, the baselength is the number of seconds covered by the moving average. This baselength is also used for band-pass filtering.

7. **Filter using original data** check box: check to apply the specified filters to the original data—the values read from the file. When not checked, the filters are applied to the data as modified by previous filtering. If this box is not checked, the data are processed every time the filter dialog box is displayed and exited with the **OK** button.

8. **Help** button. Click to display the following information.


Filter Help
Original Data
After Low-pass Filter
After High-pass Filter
Low-Pass Filter
A low-pass moving average filter transforms a series of numbers by replacing each number with the average value, where the average is taken over some number that corresponds to a base length. The number of points used in the average is the baselength, divided by the interval between points. The averaging process smooths the data, removing high frequencies and passing low frequencies.
High-Pass Filter
A high-pass moving average filter works by applying a low-pass smoothing filter, and then subtracting the smoothed points from the original, leaving deviations from the smoothed data. This removes low frequecies,

9. **Cancel** button. Click to exit the dialog box without changing the data in the active plot.

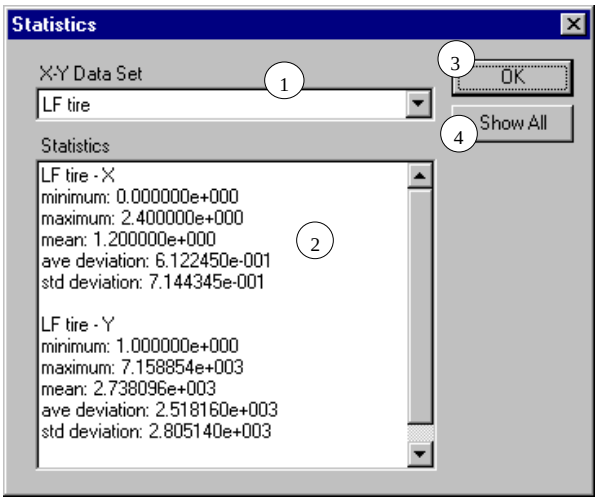
10. **OK** button. Click to exit the dialog box and apply the specified filter to all Y variables in the active plot.

Statistics...

Select this command to display a dialog box with simple statistics for the plotted variables. The numbered items in the dialog box are described below.

1. X-Y data set name. Use the pull-down menu to pick the data set in the active plot whose statistics will be displayed.

2. Statistics for specified data set. The field is sized to display the statistics for the two variables making up a single X-Y data set. If you choose to show the statistics for all data sets in the plot, then the scroll bar can be used to view the statistics for the data sets after the first.



(3) **OK** button. Click to exit the dialog box.

(4) **Show All** button. Click to display the statistics of all data sets in the active plot.

Labels...

Use this command to display the **Labels** dialog box. The keyboard command is Ctrl+L. Use this dialog box to edit the plot title, X axis, Y axis, and legend labels for the active plot. Figure 7.7 shows the dialog box along with an example active plot. The circled numbers appear both in the plot and the dialog box, to indicate the correspondence between the field in the dialog box and the plot.

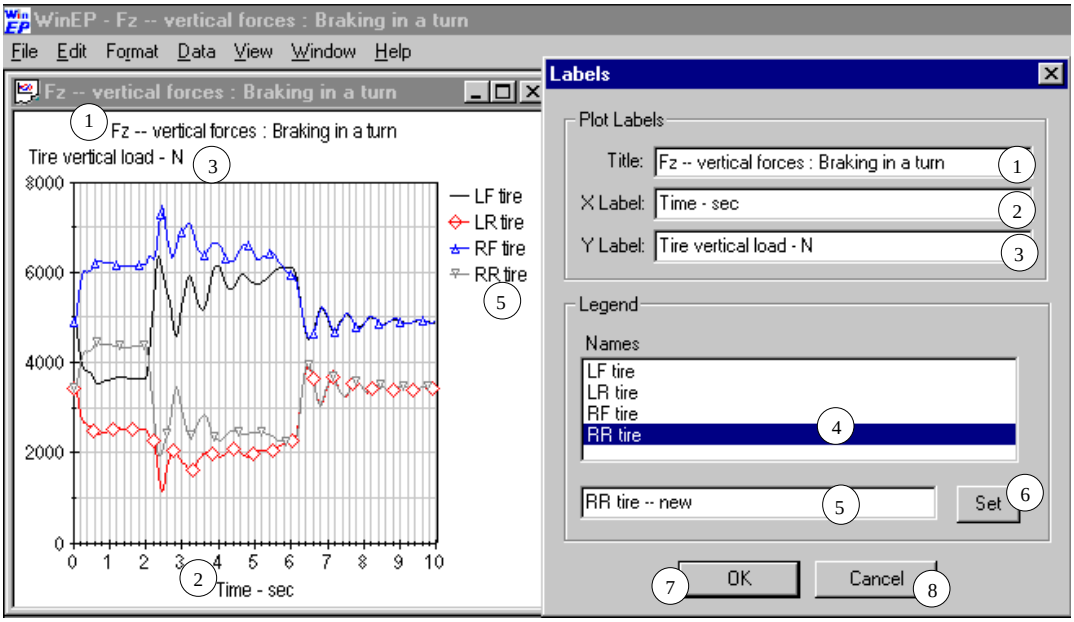


Figure 7.7. Labels dialog box.

(1) Title. The title is shown in the top of the plot and also in the title bar of the plot window. It can be edited in the dialog box.

(2) X Label. This is the label written underneath the X axis. It can be edited in the dialog box.

(3) Y Label. This is the label written above the Y axis. It can be edited in the dialog box.

(4) Names list, used to identify the X-Y data sets in the legend. To change any of these labels:

a. Select it from the Names list (4).

b. Edit the name as it appears in the edit field (5).

c. Click the Set button (6) to apply the change.

(5) Edit field used to change a name in the selected legend.

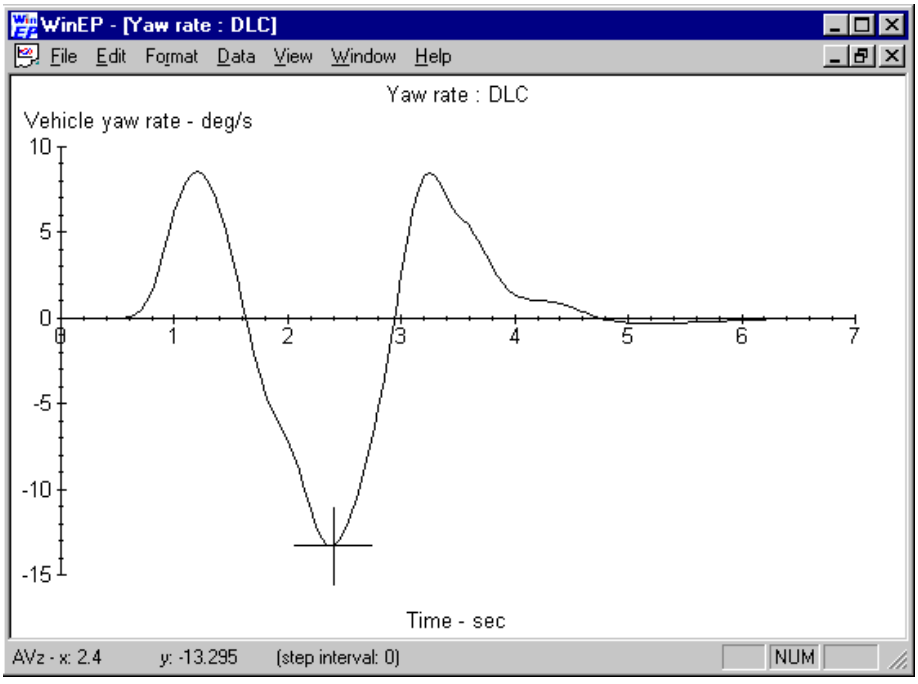
(6) **Set** button. Click to set the name selected in the Names list (4) to match the text in the edit field (5).

(7) **OK** button. Click to close the dialog box and apply the new settings to the active plot.

(8) **Cancel** button. Click to close the dialog box without modifying the active plot.

Show Data Points

This command toggles the mode of showing a cursor on the plot and the associated coordinates in the status bar. The keyboard command is Ctrl+D.



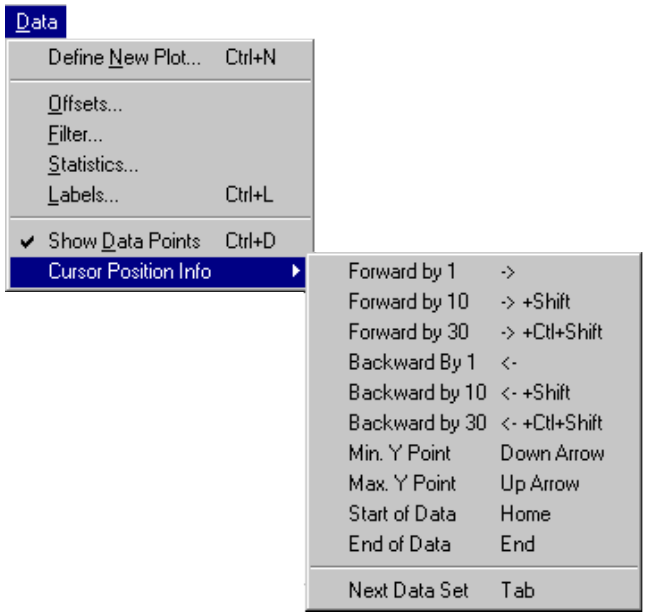
When you select this option, a cursor will appear at the first data point contained in the data file. For time history plots, the first point is at the far left side of the plot. However, for cross-plots, the first point may be located somewhere else.

The X and Y values associated with the cursor are displayed in the status bar. For example, in the figure below, the cursor is at the point: time = 2.4 sec, Yaw velocity = –13.295 deg/s.

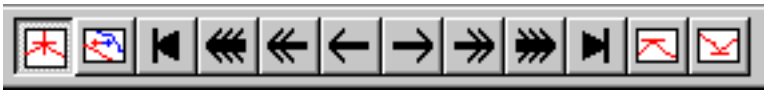
> Note: When there are multiple plot windows, the cursor and X-Y values are always associated with the active window.

Cursor Position Info

This menu item displays a sub menu whose main purpose is to remind you of keyboard commands that move the cursor. Although the menu items are functional, you will usually find it quicker to press the indicated keys.



All of the functions on the sub menu are also available as buttons on the tool bar. The buttons are dimmed unless the option to **Show Data Points** is enabled. This is indicated visually in the toolbar by the button being “pushed in” as shown below.



Forward by 1 — use the right arrow key (→) to move one point forward, based on the order the X values are stored in the file. When the cursor is on the last point in the file, this command has no effect.

Forward by 10 — use the right arrow key (→) plus the Shift key to move 10 points forward, based on the order the X values are stored in the file. When the cursor is on the last point in the file, this command has no effect.

Forward by 30 — use the right arrow key (→) plus the Shift key plus the Control key to move 30 points forward, based on the order the X values are stored in the file. When the cursor is on the last point in the file, this command has no effect.

Backward by 1 — use the left arrow key (←) to move one point backward, based on the order the X values are stored in the file. When the cursor is on the first point in the file, this command has no effect.

Backward by 10 — use the left arrow key (←) plus the Shift key to move 10 points backward, based on the order the X values are stored in the file. When the cursor is on the first point in the file, this command has no effect.

Backward by 30 — use the left arrow key (←) plus the Shift key plus the Control key to move 30 points backward, based on the order the X values are stored in the file. When the cursor is on the first point in the file, this command has no effect.

Min Y Point — use the down arrow key (↓) to locate the point with the minimum Y value for the points currently displayed.

Max Y Point — use the up arrow key (↑) to locate the point with the maximum Y value for the points currently displayed.

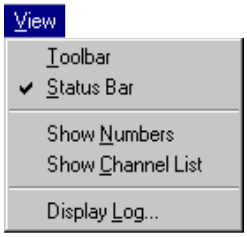
Start of Data — use the `Home` key to find the first point. This is the same as pressing the left arrow repeatedly.

End of Data — use the `End` key to find the last point. This is the same as pressing the right arrow repeatedly.

Next Data Set — use the Tab key to move the cursor between currently displayed data sets in overlay plots. If the active plot has only a single data set, then this command has no effect.

View Menu

Use this menu to toggle the display of the status bar, the tool bar, and to generate information about the data files.



Toolbar

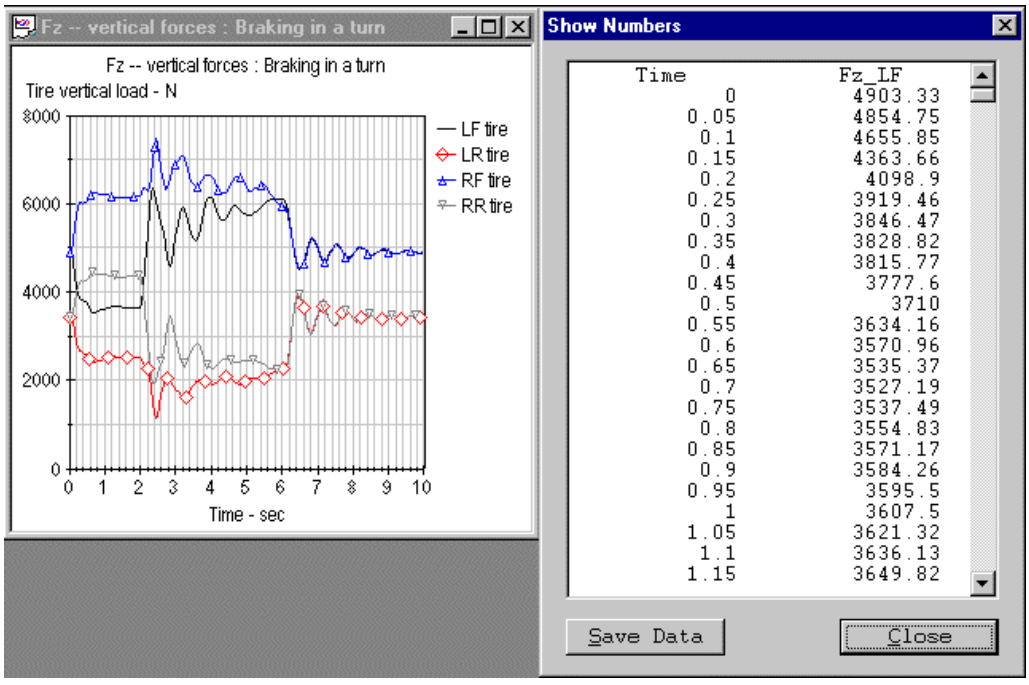
Toggle this item to display or hide the toolbar.

Status Bar

Toggle this item to display or hide the status bar at the bottom of the WinEP work space.

Show Numbers

Select this item to view the currently plotted data in tabular form. Numbers shown in the window can be copied to the clipboard, and the entire contents can be saved to file by using the **Save Data** button.



If the plot includes many numbers, it can take some time for the window to appear.

The same function can be performed using the **File** menu command to **Save Data**.

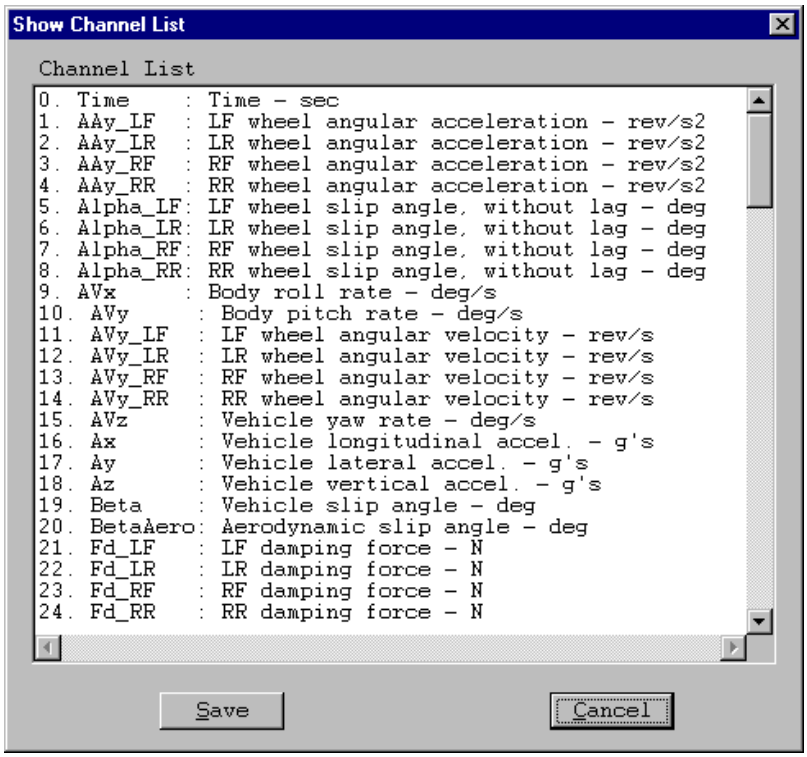
> Note: Sometimes, due to memory limitations, only some of the values will appear in the list. When this happens, use the **Save Data** button described in the next paragraph.

Show Channel List

Select this item to view all variable names from the current ERD file. Unlike the short names shown in the **Channel Select** dialog box (see Figure 7.6), these names are longer and include the units.

Text shown in the window can be copied to the clipboard, and the entire contents can be saved to a file by using the **Save** button.

The same function can be performed using the **File** menu command to **Save Channel List**.

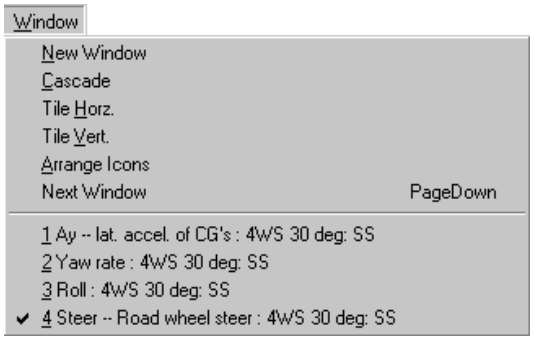

Show Channel List
Channel List
0. Time : Time - sec
1. AAy_LF : LF wheel angular acceleration - rev/s2
2. AAy_LR : LR wheel angular acceleration - rev/s2
3. AAy_RF : RF wheel angular acceleration - rev/s2
4. AAy_RR : RR wheel angular acceleration - rev/s2
5. Alpha_LF: LF wheel slip angle, without lag - deg
6. Alpha_LR: LR wheel slip angle, without lag - deg
7. Alpha_RF: RF wheel slip angle, without lag - deg
8. Alpha_RR: RR wheel slip angle, without lag - deg
9. AVx : Body roll rate - deg/s
10. AVy : Body pitch rate - deg/s
11. AVy_LF : LF wheel angular velocity - rev/s
12. AVy_LR : LR wheel angular velocity - rev/s
13. AVy_RF : RF wheel angular velocity - rev/s
14. AVy_RR : RR wheel angular velocity - rev/s
15. AVz : Vehicle yaw rate - deg/s
16. Ax : Vehicle longitudinal accel. - g's
17. Ay : Vehicle lateral accel. - g's
18. Az : Vehicle vertical accel. - g's
19. Beta : Vehicle slip angle - deg
20. BetaAero: Aerodynamic slip angle - deg
21. Fd_LF : LF damping force - N
22. Fd_LR : LR damping force - N
23. Fd_RF : RF damping force - N
24. Fd_RR : RR damping force - N
Save
Cancel

Display Log...

This displays an error message window sometimes used for debugging.

Windows Menu

Use this menu to control the appearance of the plot windows in WinEP.

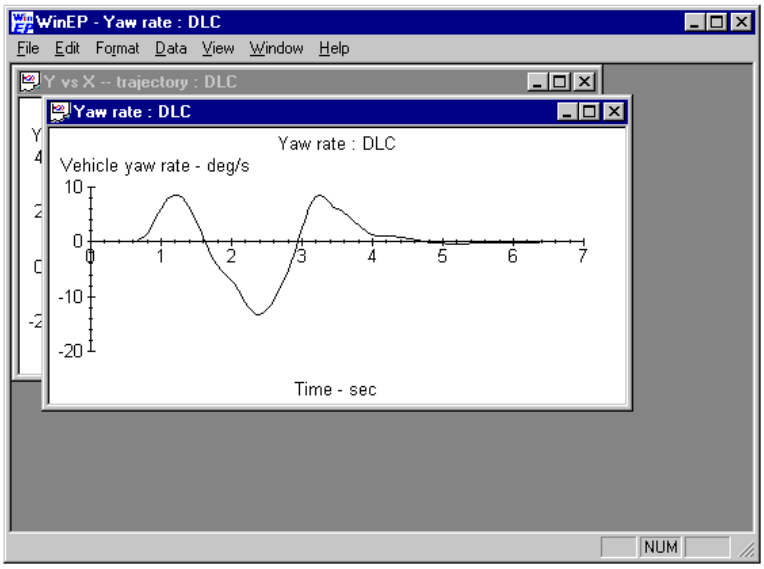

Window
New Window
Cascade
Tile Horz.
Tile Vert.
Arrange Icons
Next Window PageDown
1 Ay -- lat. accel. of CG's : 4WS 30 deg: SS
2 Yaw rate : 4WS 30 deg: SS
3 Roll : 4WS 30 deg: SS
4 Steer -- Road wheel steer : 4WS 30 deg: SS

New Window

Use this command to create a new window with the data and all format settings from the currently active window. The new window will be displayed in front of the other windows.

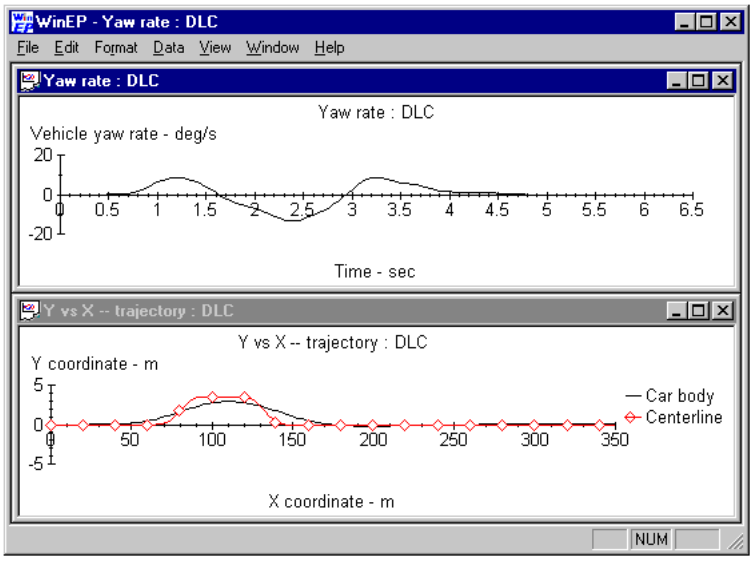
Cascade

Use this command to arrange all plot windows in a cascade, as shown below for two windows.



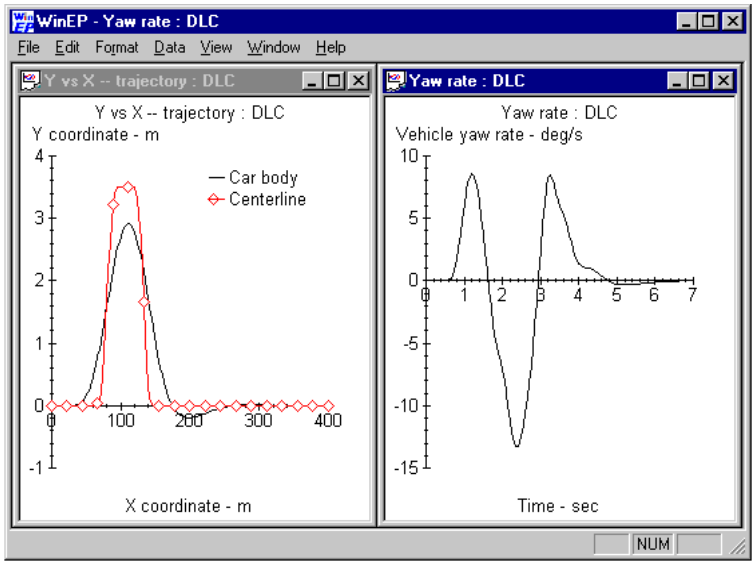
Tile Horz.

Use this command to tile the plot windows with horizontal divisions, as shown below for two windows. If there are four or more windows, they are arranged in columns and rows.


WinEP - Yaw rate : DLC
File Edit Format Data View Window Help
Yaw rate : DLC
Yaw rate : DLC
Vehicle yaw rate - deg/s
20
0
-20
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5
Time - sec
Y vs X -- trajectory : DLC
Y vs X -- trajectory : DLC
Y coordinate - m
5
0
-5
0 50 100 150 200 250 300 350
Car body
Centerline
X coordinate - m
NUM

Tile Vert.

Use this command to tile the plot windows with vertical divisions, as shown below for two windows. If there are four or more windows, they are arranged in columns and rows.



Arrange Icons

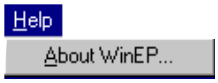
Use this command to tidy any “iconized windows” in the WinEP workspace.

Next Window

Use this command to change the active plot window. The keyboard command is the PageDown key. If there are more than two plot windows, keep pressing the PageDown key until the desired window becomes active.

Help Menu

This menu has a command to open the **About WinEP** dialog box. Use it to obtain the version number and the current web site for updates.

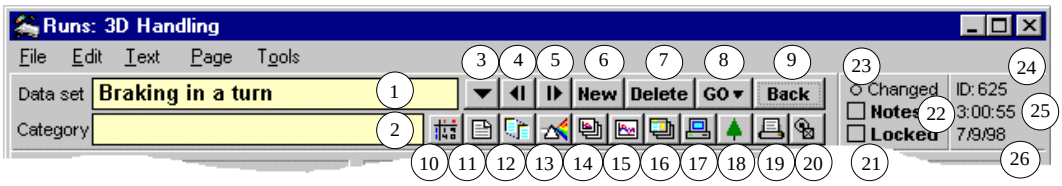


8. Design of CarSimEd Data Screens

This section describes several standard elements in the CarSimEd design that are shared in many of the data screen. All data screens have the same set of controls at the top in a ribbon bar. All have the same menu bar. Also, data screens with tabular data have a common layout.

The Ribbon Bar

Almost every screen in CarSimEd has a standard ribbon bar at the top. It contains the name of the data sets, information about the last time the data set was modified, and a number of navigational controls.



Fields

(1) Data Set field. This field shows text entered by you, the user, to name the current data set. There are a few restrictions on titles: (1) each data set in the library must have a unique title; (2) the title cannot include a comma (the software will automatically replace it with a semicolon); and (3) the length of the title is limited to 32 characters.

> Note: On the **Runs** screen this is where you give a unique name to the run of a specific vehicle, maneuver, and simulation model. In a vehicle screen, it typically serves to identify the vehicle and its properties, and so on for other screens.

(2) Category field. This field shows text you use to create a sub-menu for listing the contents of the library. The Category field is provided as a convenience for you to divide large numbers of data sets into logical groups in pull-down menus, as shown in Figure 8.1. If the category field is blank then the title is shown in the top-level menu.

> Note: Data sets in different categories are still subject to the restriction that they cannot have the same names.

Buttons

(3) Library triangle button. Press this button to display a pull-down menu listing of all of the data sets in the library, grouped by categories.

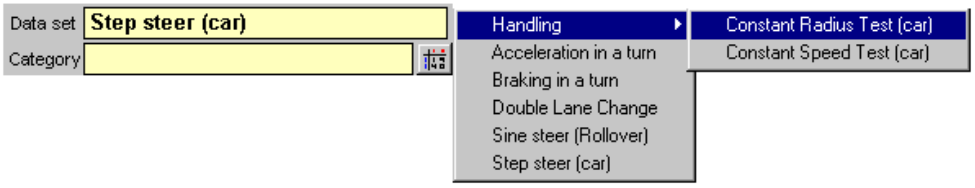


Figure 8.1. Pull-down menu by the Data Set field.

(4) Library `Left` button. The data sets in the library are ordered alphabetically first by category and then by title. Click this button to go to the preceding data set. If the current data set is the first one, this button takes you to the last one (the ordering is circular).

(5) Library `Right` button. The data sets in the library are ordered alphabetically first by category and then by title. Click this button to go to the next data set. If the current data set is the last one, this button takes you to the first one (the ordering is circular).

(6) **New** button. Click this button to make a full copy of the data set. The copy is identical to the original, except that the name (1) will be different (usually it will have a number appended to it). When the copy is made, the Data Set field will be highlighted to encourage you to type a distinctive title for the new data set.

(7) **Delete** button. Click this button to delete the data set.

When you click the **Delete** button, two things happen: (1) the current data set as displayed on the screen is deleted, and (2) auxiliary files associated with the data set are also deleted. Thus, CarSimEd automatically performs housecleaning.

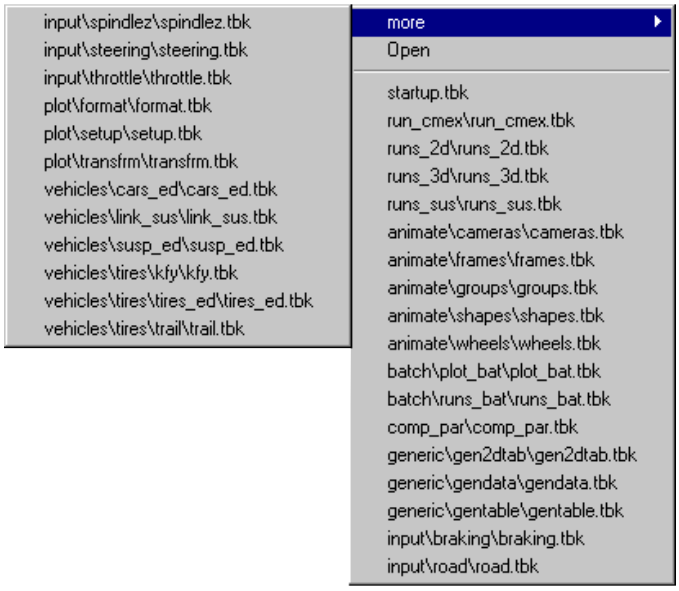
If the library has only one data set, the **Delete** button cannot be used — there must always be a minimum of one data set in a library. Also, the **Delete** button will not work on the current data set if the **Locked** box (21) is checked.

There are two keyboard modifiers for the **Delete** button:

- Press the `Ctrl` key when clicking the button to avoid the confirmation message.
- Press the `Shift` key when clicking the button to delete all data sets except the current one. There will be a confirmation message, giving you a chance to change your mind after clicking the button. Be aware that if you go ahead, other data sets can be deleted even if their **Locked** boxes are checked. **Use this feature with care!**

> Note: When you position the cursor over the **Delete** button, the Status Bar at the bottom of the window describes the above two keyboard modifications.

⑧ **GO** button. Press this button to display a menu of all libraries in CarSimEd. Select a library from this menu to leave the current library and go to the selected one.



The menu contains more items than will fit on a monitor with VGA resolution (640 x 480). Note that the first item in the menu is **More**. Highlight it to view more menu items, as shown in the figure.

⑨ **Back** button. Click this button to go back to the previous library. This is the library that you came from by using a link, the GO menu, or the History dialog box.

⑩ **Number Tools** button. Click this button to go to the calculator screen. This is equivalent to the **Tools** menu **Calculator** command. The calculator screen is used to create and edit tabular data. If it is not currently linked, control-click to get a file browser to locate the library and establish a link to it. The default is `Sgui_lib\Calc.tbk`.

⑪ **Text Editor** button. Click this button to go to a utility library used to create and edit tabular data. This is equivalent to the **Tools** menu **Text Editor** command. This can also be set to go to a text editor, such as the Windows **Notepad** program. If it is not currently linked, control-click to get a file browser to locate an executable program file and establish a link to it. The default link is `Programs\WinVI.exe`.

⑫ **Library Editor** button. Click this button to bring up a floating window that can be used to edit libraries. This is equivalent to the **Tools** menu **Library Editor** command. The current library is automatically loaded into the editor window.

⑬ **Spectrum Analyzer** button. Click this button to go to the **Spectrum Setup** library with settings for a spectrum analyzer. This is equivalent to the **Tools** menu **Spectrum** command. Control-click to get a file browser to make a different library the default.

> Note: The spectrum analyzer and the settings screen are not present in the standard CarSimEd package.

⑭ **Batch Plot** button. Click this button to go to the **Plot Setup: Batch** library. This is equivalent to the **Tools** menu **Batch Plotting** command. Control-click to get a file browser to make a different library the default. The default link is `Batch\Plot_bat\Plot_bat.tbk.`

⑮ **Plot Setup** button. Click this button to go to the **Plot Setup: Single** library with plot settings that define plot variables, formats, etc. This is equivalent to the **Tools** menu **Plot Setup** command. Control-click to get a file browser to make a different library the default. The default link is `Plot\Setup\Setup.tbk.`

⑯ **Batch Runs** button. Click this button to go to the **Runs: Batch** library. This is equivalent to the **Tools** menu **Batch Runs** command. Control-click to get a file browser to make a different library the default. The default link is `Batch\Runs_bat\Runs_bat.tbk.`

⑰ **Runs** button. Click this button to go to a **Runs** library with settings to make a single simulation run. This is equivalent to the **Tools** menu **Runs** command. Control-click to get a file browser to make a different library the default. The default link is `Runs\Runs.tbk.`

⑱ **Parstree** button. Click this button to run the Parstree program for viewing the contents of the PAR file associated with the current data set, plus the PAR files for all of its linked data sets. For example, Figure 8.2 shows the display for a data set from the **Runs** library. The file 94.par is selected in the left pane, and the contents of that file are shown in the right pane.

If it is not currently linked, control-click to get a file browser to locate an executable program file and establish a link to it. The default link is `Programs\Parstree.exe`.

⑲ **Print** button. Click this button to print a copy of the current screen display.

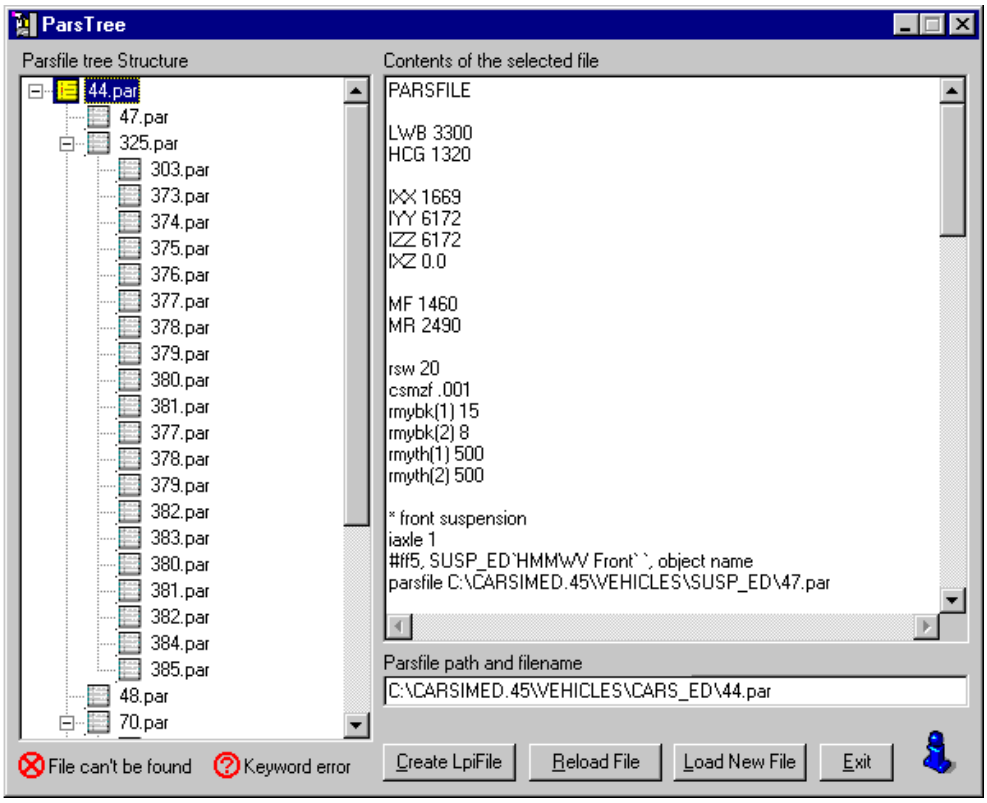
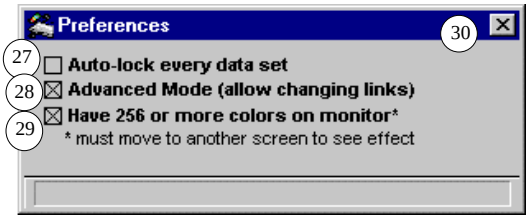


Figure 8.2. The Parstree window displayed by clicking the Parstree button.

(20) **Preferences** button. Click this button to bring up a floating window used to set global preferences for CarSimEd. This is equivalent to the **Tools** menu **Preferences** command.



(27) **Auto Lock**. When checked, CarSimEd automatically locks every data screen when you leave it. Otherwise, data sets are locked only when you click the locked boxes (21) manually.

(28) **Advanced Mode**. When checked, you can change links to connect with different libraries. When not checked, you must live with the links the way they are.

(29) **Have 256 or more colors on monitor**. Check this box if your video display supports 256 or more colors. Un-check it if your display is limited to 16 or fewer colors.

(30) **Close**. Click here to exit the **Preferences** window and apply the current preference settings.

Check Boxes

(21) **Locked** box. Click this box to check or un-check it. When checked, the data set is locked and you cannot change text in any of the edit fields or any links. You also cannot delete the data set using the **Delete** button (unless you simultaneously press the `shift` key).

(22) **Notes** box. Click this box to check or un-check it. When checked, the **Notes** field is displayed. Otherwise, it is hidden. The contents of a **Notes** field are strictly for your use in documenting data. They are not sent to the solver programs.

This button hides the **Notes** field, but does nothing to the contents. You do not lose your notes when you hide them.

When you create a copy of a data set using the **New** button (6), the contents of the **Notes** field are also copied, even if the box is not checked.

Data Set Information

(23) Changed circle. The small circle is normally transparent. However, a red dot appears when a link is changed, or when a field is changed and you leave the field by clicking somewhere else, using the `Tab` key, etc. The red dot indicates that the PAR file associated with the current data set will be updated when you leave the screen.

Click on the red circle to force CarSimEd to overwrite the PAR file immediately, such as when testing new animator settings as described in section **Testing Animator Data Sets** in Chapter 5 on page 77.

(24) ID Number. Each data set in the library has a unique ID number. The number is used to automatically name the PAR files associated with the data set. For example, if the ID is 131, a file `131.par` is created automatically by CarSimEd and located in the same folder as the `TBK` library file.

(25) Time of last change. This is set whenever the PAR file is updated.

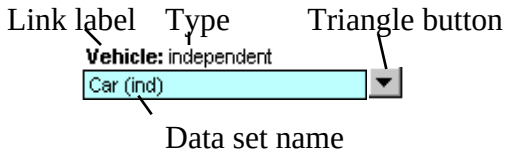
(26) Date of last change. This is set whenever the PAR file is updated.

Data Links

Each CarSimEd screen contains a data set that is part of a relational database. Data sets are linked together approximately in a top-down hierarchical fashion as shown earlier in Figure 3.4 on page 26.

Parts of a Data Link

The object used to link data sets together is called a data link. As shown below, a data link has several components:



- The name of the linked data set is shown in the blue field, e.g., `Car (ind)`.

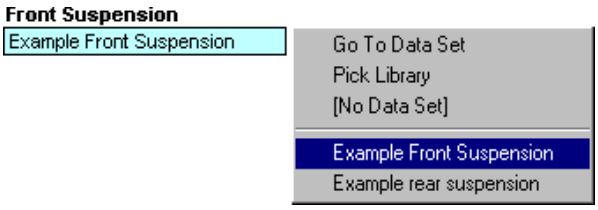
> Notes: `{No library selected}` appears in the blue field if the link is not pointing to anything.
>
> {No data set selected} appears if the link is pointing to a library, but has not yet been linked to a specific data set in that file.

- Most data links have labels, in bold typeface, that describe the type of data to which the link points, e.g., **Vehicle**. In some cases, additional information is shown in plain typeface, e.g., `independent`.
- The triangle button is used to display a pull-down menu.

There are two keyboard modifiers for the triangle button:

1. Press the Ctrl key when clicking the button to get the full pathname of the linked library, e.g., `C:\CarSimEd\Vehicles\Vehicles\Vehicles.tbk`.
2. Press the Shift key when clicking the button as a shortcut for selecting the menu item **Go To Data Set**. This will take you to the linked data set (it is the same as selecting the first menu item). This is usually quicker than using the menu.

The Data Link Menu

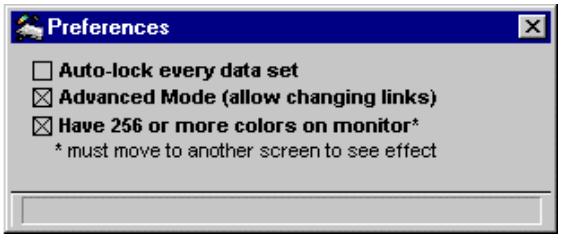


The items of the data link menu perform the following functions:

- **Go To Data Set**. Goes to the currently linked data set.
- **Pick Library.** Brings up the file browser dialog box to select a new TBK library file for the link. The browser may request a library of a certain type, e.g., **Vehicle**. If the TBK file that is selected does not contain the right kind of data, a warning is displayed. However, you can override the warning and make the link anyway. **This is not recommended for beginning users.**

  This menu item is dimmed when the preferences for CarSimEd are set with the Advanced mode disabled. If you want to pick a new library and the menu command is dimmed, you must go to the preferences window by clicking the

preferences button in the ribbon bar ( ) or selecting the **Preferences** command from the **Tools** menu. This brings up a dialog box:


Preferences
Auto-lock every data set
Advanced Mode (allow changing links)
Have 256 or more colors on monitor*
* must move to another screen to see effect

Make sure the **Advanced Mode** box is checked and then close the preferences window.

- **[No data set selected].** This breaks the current link to a data set. The link to the library remains, but, without a data set, data from the library are not used in simulations.
- *Data Set Names.* All of the menu items after the dividing line are the names of the data sets in the linked library. Change the link to a different data set by selecting its name.

If the current screen has the **Locked** box checked, only the Go To Data Set item is active. The other items involve changes in the data set and are not allowed.

Tabular Data

Some CarSimEd data sets involve tables of numbers. The values are plotted to show the functional relation graphically and to help identify errors. Figure 8.3 shows an example data set.

> Note: For these data sets, the **Notes** field is often located on top of the table of numbers (2). Therefore, to view the numbers in the table, the **Notes** button must not be checked.

There are four standard items on each screen with tabular data.

(1) Plot. The plot is based on the tabular data (2), using the first number on each line for an X value and the second number for a Y value. The axes are scaled automatically to include the full range of numbers in the table.

Except when the data set is locked, when the mouse pointer is positioned over the plot, the corresponding X and Y values are shown in the status bar. Clicking the left mouse button causes the currently displayed X and Y values to be appended to the table. This feature can be used to rapidly build a new data set. The new data points are not plotted until the **Update Plot** button (3) is clicked.

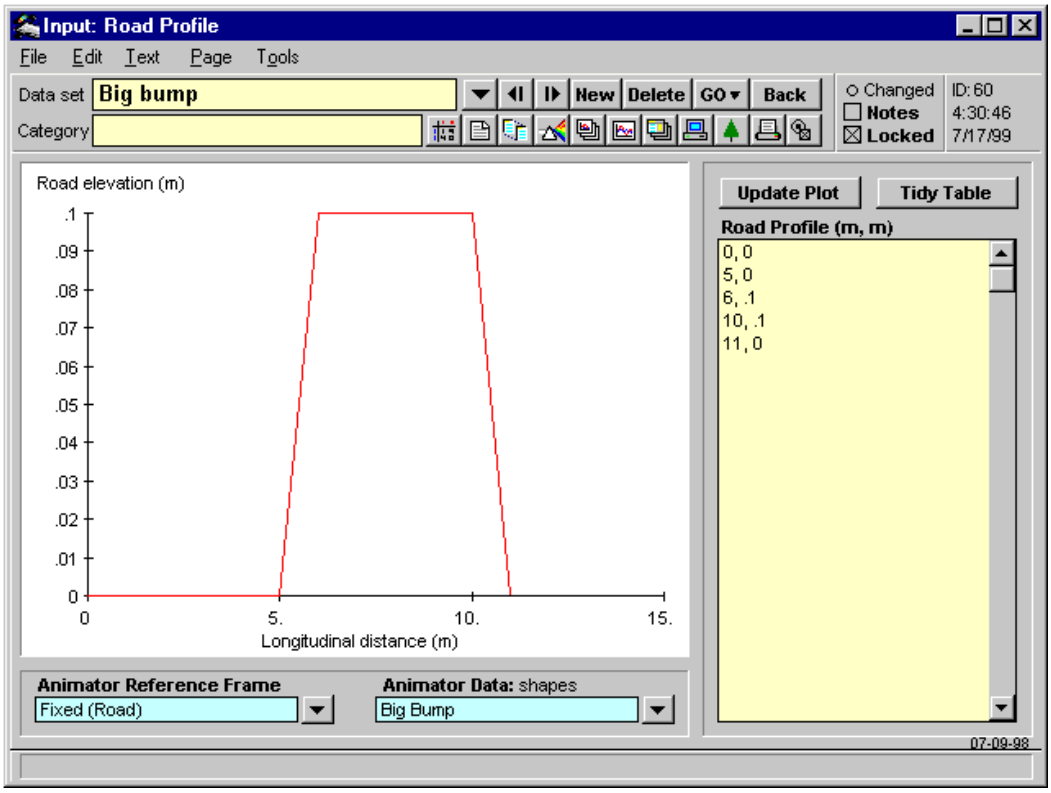


Figure 8.3. Example screen for tabular data.

(2) Table field. Each line in this field should have an X value followed by a Y value. The numbers must be separated by a comma. Additional information can be added after a second comma. These numbers are passed to the solver programs exactly as they appear in the field.

> Note: The table field requires at least two lines of data in order to create a plot (1). If you attempt to make a plot or leave the screen with only one pair of X-Y points, an error message is generated.

The plot (1) shown on this type of screen covers only the range of data. In the simulation, values might extend beyond the range shown. For example, if the simulation runs longer than 210 seconds, the independent variable (time) would have a value outside the range covered by the table.

The solver programs in CarSimEd have two methods for extrapolating outside the range of a table.

- For variables that are inputs to the vehicle (controls, such as steering wheel input, or disturbances, such as wind), *flat-line extrapolation* is used. If the independent variable is less than the first value in the table, the first value of the dependent value is used. If the independent variable is greater than the last (highest) value in the table, then the last value of the dependent variable

is used. For example, the data shown in Figure 8.3 would cause a steering wheel angle of 0.0 to be used for values of time less than zero. For values of time greater than 210 seconds, the steering wheel angle would be 230 degrees.

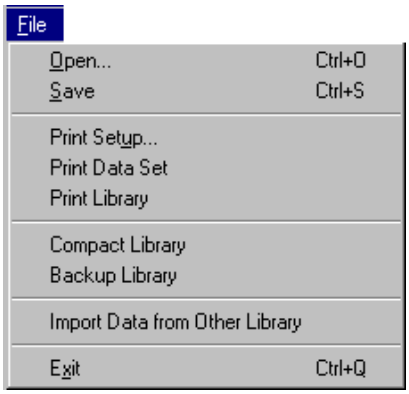
- For variables that describe vehicle properties (for example, spring force vs. deflection), *constant-grade extrapolation* is used. Depending on whether the independent value is less than or greater than the range of the table, the first or last two points are used for extrapolation by assuming their gradient for values outside the defined range.

(3) **Update Plot** button. Click this button to create a new plot of the data in the field (2).

(4) **Tidy Table** button. Click this button to "clean up" the appearance of the numbers in the table field. A space is inserted after the comma, and additional information after a second comma is removed. This operation is useful when tabular data are pasted into the edit field from some other source, such as an echo file created by a CarSimEd solver program.

File Menu

The File menu contains commands pertaining to library TBK files. In CarSimEd, this menu is not used as much as in most Windows programs, because the opening and saving of files is handled automatically.



Open…

Select this menu item to open an arbitrary TBK library file. The keyboard command is Ctrl-O.

Save

Select this menu item to immediately save the current TBK file to disk. The files are automatically saved whenever you move from one library to another, so this command is used mainly when the machine stability is in doubt and you fear a program crash. The keyboard command is Ctrl-S.

Print Setup…

Select this menu item to select a Windows printer and optionally configure it.

Print Data Set

Select this menu item to print the current screen.

Print Library

Select this menu item to print all data sets in the current library, using the data screen layout.

Compact Library

This command causes ToolBook to compact the file, eliminating space that was used by data sets that have been deleted.

Over time, the TBK library files can grow more than expected, because when you delete data sets the file space is not recovered. Use this menu item if you notice that some of the TBK files have gotten very large. Depending on the amount of graphic content, the normal size of a TBK file can be as small as 70K, or as large as 700K.

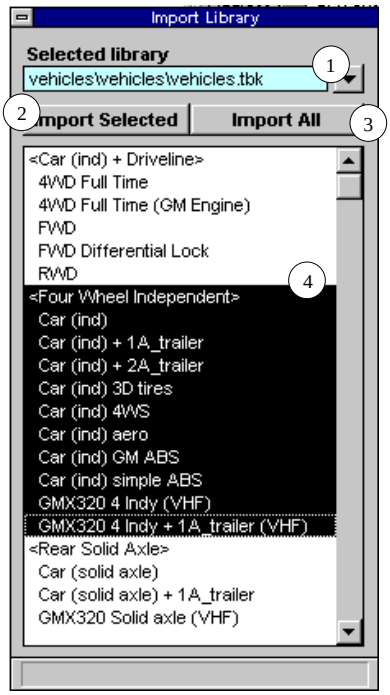
Backup Library

Select this menu item to make a backup copy of the current TBK library file. The copy is given the extension BAK.

Import Data from Other Library

Select this menu item to import one or more data sets from another TBK file into the current one. This command is useful for transferring information between separate CarSimEd installations. It is also useful for importing data from an old version of CarSimEd into an updated version.

When the command is selected, the Windows File browser appears with a request to locate a TBK file with data to import. If a file is selected, the contents are displayed in the following dialog box. The numbered items are described below.



(1) Link to the selected "source" TBK file. The adjacent button can be used to select a different file.

(2) **Import Selected** button. Click this button to import the data sets selected in the field (4). Each data set in the source library is copied into the currently open "target" library. If a target data set already exists with the same name, it is given a unique name by appending # and a number.

The importing process involves taking every record field (yellow or blue) that exists in both the source and target data sets, and copying the values from the source to the target. If a field does not exist in the source, the value shown in the current data set in the target is unchanged. If the field does not exist in the target, the value is not copied.

(3) **Import All** button. Click this button to import all data sets from the linked file (shown in (1)). The process is the same as described above for the Import Selected button (2).

> Note: If you select more than 10 data sets to import, a warning message appears warning of possible problems. Due to a low-level interaction between ToolBook and Windows, ToolBook will sometimes crash while trying to import many data sets. It is recommended that you import less than 10 data sets at a time, and that you back up the library file that is the target of the import process.

(4) List of data sets in the linked library indicated in (1).

Exit

Select this menu item or use the keyboard command Ctrl+Q to exit from CarSimEd.

Edit Menu

The **Edit** menu supports the clipboard and the **Find** command.

Undo

Select this menu item to undo the most recent entry or change in a text field. The keyboard command is Ctrl+Z.

Cut

Select this menu item to cut the currently selected text to the clipboard. The keyboard command is Ctrl+X.

Copy

Select this menu item to copy the currently selected text to the clipboard. The keyboard command is Ctrl+C.

Paste

Select this menu item to paste the clipboard to the current cursor location. If text is currently selected, it is replaced by the contents of the clipboard. The keyboard command is Ctrl+V.

Clear

Select this menu item to clear the currently selected text. The keyboard command is the delete key.

Select All

Select this menu item to select all text in the field where the cursor is currently located. The keyboard command is Ctrl+A.

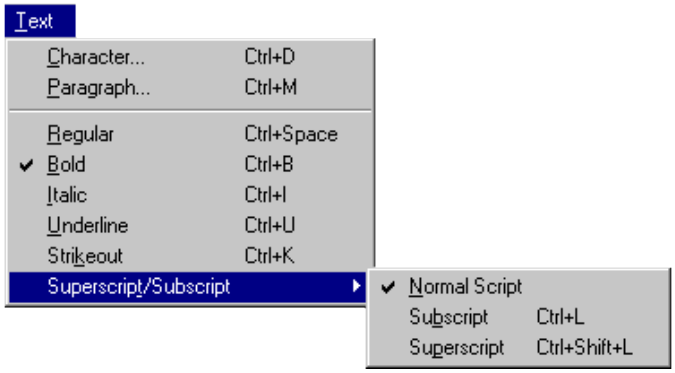
Find…

This command can be used by advanced users to locate information in CarSimEd data fields.

> **WARNING**: While you are using this option it is possible to trick the underlying ToolBook runtime system, leading to errors that will corrupt your files. Use this option with caution.

Text Menu

The **Text** menu contains commands involving the display of text in edit fields and notes. Although text formatting has no effect on how the simulations are run, you can control the appearance of the text in notes and data fields if you wish.



> Note: The items in the **Text** menu apply only to the currently selected text in a single field.

Character

Select this menu item to bring up a floating window to change the character properties of the selected text (Font, Font Style, Size, etc.). The keyboard command is Ctrl+D.

Paragraph

Select this menu item to bring up a floating window to change the spacing (e.g., Alignment, Indentation, Spacing, etc.) of the selected text. The keyboard command is Ctrl+M.

Regular

Select this menu item to remove any special character properties of the selected text, such as bold, underline, etc. The keyboard command is Ctrl+Space.

Bold

Select this menu item to toggle the selected text between **bold** and not bold. The keyboard command is Ctrl+B.

Italic

Select this menu item to toggle the selected text between *italic* and not italic. The keyboard command is Ctrl+I.

Underline

Select this menu item to toggle the selected text between underline and not underline. The keyboard command is Ctrl+U.

Strikeout

Select this menu item to toggle the selected text between ~~strikeout~~ and not strikeout.

Normal Script

Select this menu item to remove subscript/superscript formatting. The keyboard command is Ctrl+K.

Subscript

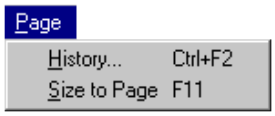
Select this menu item to toggle the selected text between $_{\text{subscript}}$ and not subscript. The keyboard command is Ctrl+L.

Superscript

Select this menu item to toggle the selected text between $^{\text{superscript}}$ and not superscript. The keyboard command is Ctrl+Shift+L.

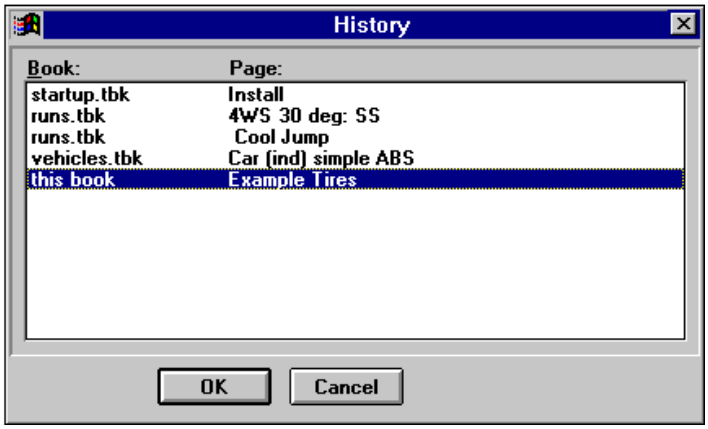
Page Menu

This menu offers access to previously viewed screens (pages), and also control of how the current data screen is viewed.



History

Select this menu item to bring up the following floating window to see a list of the libraries that you have visited during this session. From the window you can select a library and go to it by double-clicking on it or clicking the OK button.

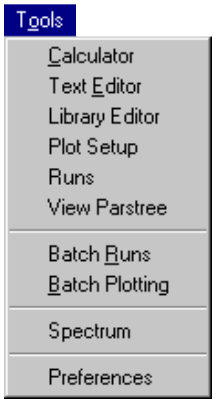


Size to Page

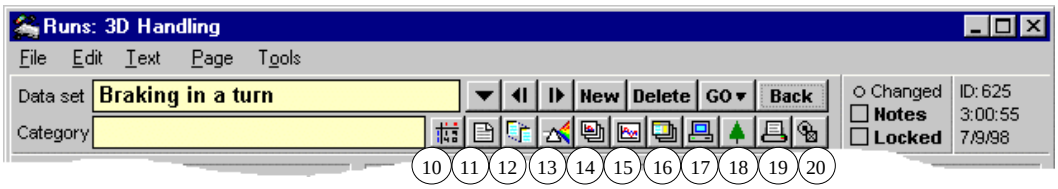
Select this menu item to set the CarSimEd window to its default size. If you happen to click on the Windows zoom icon, Windows will fill your screen with the current window. However, the window doesn't show any more information. Use this menu item to restore its appearance.

Tools Menu

Use the Tools menu as an alternative to the buttons in the ribbon bar.



The ribbon bar, shown below, was described in more detail starting on page 110.



Calculator

Select this menu item to go to the calculator screen (same as (10)).

Text Editor

Select this menu item to go to a utility library used to create and edit text files (same as (11)).

Library Editor

Select this menu item to bring up a floating window used to edit existing libraries (same as (12)).

Plot Setup

Select this menu item to go to the **Plot Setup: Single** library with plot settings that define plot variables, formats, etc. (same as (15)).

Runs

Select this menu item to go to a **Runs** library with settings to make a single simulation run (same as (17)).

View Parstree

Select this menu item to run the Parstree program and view the tree of PAR files that starts from the current data set (same as (18)).

Batch Runs

Select this menu item to go to the **Runs: Batch** library (same as (16)).

Batch Plotting

Select this menu item to go to the **Plot Setup: Batch** library (same as (14)).

Spectrum

This item is reserved for the **Spectrum Setup** library (same as (13)).

Preferences

Select this menu item to bring up a floating window used to set global preferences for CarSimEd (same as (20)).

9. Alphabetical Library Reference

This chapter describes all of the libraries in CarSimEd. They are listed in alphabetical order, based on the screen (window) title.

Conventions in This Chapter

Each section shows an example data screen and describes its data fields, links, buttons, and other user interface objects.

Standard Objects

Standard objects that were described in Chapter 8 are not covered again in this chapter. For example, the buttons at the top of the screen (**New**, **Delete**, etc.) are the same in every library.

Keywords

Parameter values that are displayed in yellow fields will usually appear in the echo files produced by the solver programs. These echo files identify the parameters with keywords and list them in alphabetical order (see Appendix F for an example). The keywords are defined in this chapter along with the definitions for the parameters so you will know where to look for them in the echo files. They are shown in parentheses in the `Courier` font.

Many of the parameters in the CarSimEd models are applied both on the front and the rear of the vehicle. For example, the same data screen is used to describe properties of all tires, regardless of their location.

Parameters that are repeated for each axle are called indexed parameters, with the indexing being defined with the keyword `IAXLE`. The indexing keywords can be of interest to advanced users who override vehicle properties using the Miscellaneous fields scattered throughout the CarSimEd data screens. Therefore, they are specified along with the parameter keyword. For example, the entry for tire spring rate reads: Tire spring rate (keyword = `KT (IAXLE)`.

You can view the keywords in the PAR file directly by clicking the button in the ribbon bar. For example, Figure 9.1 shows some of the keywords for the **Cars** data screen.

Notice that the PARFILE shown in the figure has a couple of lines that read

```
iaxle 1
```

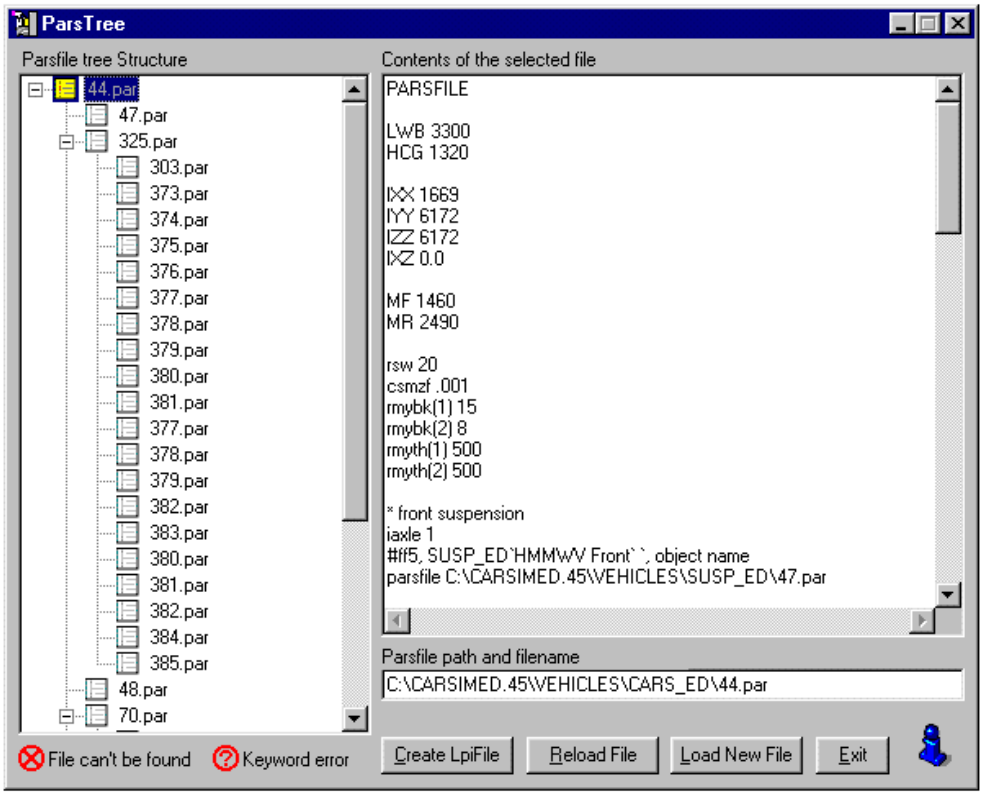


Figure 9.1. View of input PAR files using Parstree.

followed by some pathnames. This means that when any indexed parameters (including values read from other libraries) are read that involve the keyword IAXLE, they are applied to axle 1. A few lines down from the first occurrence, the IAXLE keyword is repeated with a value 2. If an indexed parameter is read again, the value is applied to axle 2.

Location of Library

Each section ends with the location of the library explained from two points of view. First is a list of the screens needed to reach the library using the user interface. For example, to get to the **Animator: Camera Setup** screen, start with the **CarSimEd Startup** screen, then press the **Start** button to go to the **Runs** screen, then follow a blue link to go to the **Animator: Camera Setup** screen. This sequence is written at the end of the reference section as:

Location in CarSimEd

CarSimEd Startup
  Runs
    Animator: Camera Setup

> Note: The names shown under **Location in CarSimEd** match the window titles and the subheading titles in this chapter. They sometimes match captions of the blue links, but not always.

In addition, a relative pathname is given for the library TBK file (e.g., `Animate\Cameras\Cameras.tbk`). These pathnames are relative to the root directory of CarSimEd. For example, the absolute pathname would be something like `C:\CarSimEd\Animate\Camera\Cameras.tbk`. The relative pathnames are useful to know even if you are not looking for a file, because they appear in the pull-down menu of the GO button (in the ribbon bar).

Animator: Camera Setup

The Animator Camera Setup screen defines the location of the virtual camera in the animator, the direction in which the virtual camera is looking, the amount of zoom, and the properties of an optional grid.

Discussion

Imagine you have a video camera and are viewing a vehicle as it moves down the road. Your location (actually, the location of the camera you are holding) determines your point of view. From that point, you can aim the camera anywhere. However, you will probably pan the camera to keep the vehicle in view. If the camera has a zoom lens, you might zoom in or out to control the size of the vehicle in the viewfinder.

In CarSimEd, there is no physical vehicle to look at. However, to go along with the simulated vehicles, the animator program described in Chapter 6 simulates the motions that you would view with a video camera. Use the **Animator Camera Setup** screen to establish various camera positions and aiming strategies.

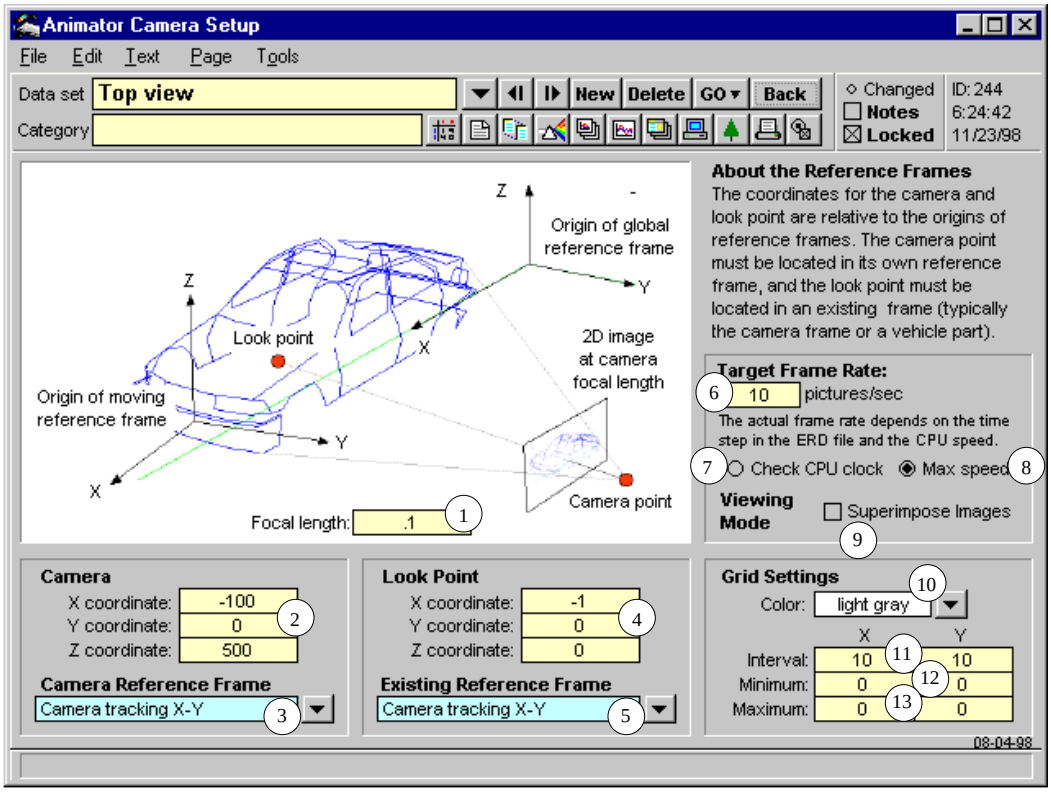
The parameters in this library are associated with two geometric points illustrated on the screen: the camera point (2), and the look point (4). At each output time interval, the animator generates a 2D image based on the relationships between the location and orientation of the simulated vehicle and the camera and look points (see the illustration on the data screen). The locations and orientations of vehicle parts are determined by the solver program. The camera and look points are determined by the information provided in this data set.

Point locations are defined in 3D space with sets of X, Y, and Z coordinates. The animator program allows both the camera and look points to be associated with any user-defined reference frame. The reference frame can be fixed or it can be a vehicle part (e.g., a sprung mass. It is even possible to define a new moving reference frame using variables that are available in the simulation output files.

Besides drawing shapes and wheels, the animator can draw a flat grid for the ground, a 3D grid for the ground, and a target path on the ground. If the simulation was run over a

3D ground surface, the 3D input to the simulation is also used by the animator. If not, the animator draws a flat grid. The grid spacing is specified on this screen. (Similar information about the target path is specified on the target path screen: **Input: Target Path For Closed-Loop Steer Control**.) The grid can be turned off manually within the Animator, as described in Chapter 6.

> Notes: Chapter 6 explains reference frames. In addition, the section **Animator Reference Frames** in this chapter describes how you define them.
>
> All of the numbers shown on this screen can be modified interactively when the animator is running. See Chapter 6 for details.



User Settings

(1) Focal length of the simulated camera (keyword = `set_focal_length`). A large value produces a telephoto lens effect and a small value produces a wide-angle lens effect. Focal length has units of meters.

(2) X, Y, and Z coordinates of the camera point (keywords = `set_camera_x`, `set_camera_y`, `set_camera_z`). These coordinates have units of meters. The coordinates apply to the reference frame specified in (3).

(3) Link to the reference frame in which the camera is located. The reference frame can be fixed or moving. The camera must have its own named reference frame (in the example above, it is `Camera tracking X-Y)`.

> Notes: The animator program requires each reference frame to have a unique name. The reference frame used to define the camera coordinates cannot be used anywhere else. If you want to position the camera in a vehicle reference frame (e.g., the sprung mass), then create a copy of the reference frame of interest, give the copy a new name, and link to the copy.
>
> Although vehicle reference frames can be used for the camera, it is more common to define a custom reference frame.

(4) X, Y, and Z coordinates of the Look Point (keywords = `set_lookpoint_x, set_lookpoint_y, set_lookpoint_z`). These coordinates have units of meters. The coordinates apply to the reference frame specified in (5).

(5) Link to the reference frame in which the look point is located. The reference frame can be fixed or moving. It is typically the same as the camera point reference frame (3). Unlike the camera reference frame, the look-point reference frame must be used somewhere else, because only the name is provided to the animator from this link. (The reference frame description is not sent to the animator, in order to avoid an error that occurs if the same reference frame name is used twice.)

(6) Target frame rate for animation—pictures per second (keyword = `set_frame_rate`). If the animation is running too slowly, you can specify a lower frame rate to speed it up. Alternatively, you can use a higher frame rate to slow it down. Typically, frame rates of 10 to 30 frames per second can run in real-time on Pentium computers. If the target frame rate is very high (more than 100), the computer will not be able to refresh the screen in real time and the animation will run in slow motion.

A high frame rate (e.g., 100) is useful for viewing wheel lock-up during brake simulations. For handling simulations, a lower frame rate ensures that the animation runs approximately in real time.

If this field is left blank, the animation runs at the default frame rate, specified under the **Preferences** option in the **Edit** menu of the animator (see Section 5).

(7) Radio button for using the computer CPU clock (keyword = set_`use_cpu_clock on`). If the animation is running faster than real time, you can select this button to slow it down to real-time. When this button is selected, the animator accesses the computer clock and waits, if necessary, to avoid running the animation faster than real time.

> Notes: The process of checking the computer clock introduces a small delay. The delay depends on the computer and how your system is set up.

> If the animation is already running slower than real-time, checking the computer clock will slow it down even more. If the animation is running slower than real time and you want to speed it up, you must decrease the target frame rate (6). Also, select button (8) to eliminate the delay caused by checking the clock.

(8) Radio button for maximum speed animation (keyword = set_`use_cpu_clock off`). When selected, the animator does not access the computer clock, allowing maximum display speed. In general you should select this box if the animation is running slower than real time on your computer.

(9) Viewing mode check box (keyword = `set_superimpose`). When this box is checked the animator does not erase each frame before drawing the next. (It superimposes images.) The values associated with the keyword (written into the text PAR file) are `on` and `off`.

(10) Grid color (keyword = `set_color`). This is text that specifies the color of the grid. Valid colors are provided in a pull-down menu.

(11) Grid intervals for X and Y (keywords = `set_interval_x`, `set_interval_y`). The units for these values are meters.

(12) Minimum X and Y values covered by the grid (keywords = `set_min_x`, `set_min_y`). The units for these values are meters. If not specified, the default values are zero.

To have the animator automatically determine the range for the grid, set the minimum and maximum values equal or leave them blank.

(13) Maximum X and Y values covered by the grid (keywords = `set_max_x`, `set_max_y`). The units for these values are meters. If not specified, the default values are zero.

To have the animator automatically determine the range for the grid, set the minimum and maximum values equal or leave them blank.

> Note: The X and Y directions are handled independently. Even if the range is set manually in one direction, the range for the other direction can be determined automatically.

Location in CarSimEd

CarSimEd Startup
  Runs
    Animator: Camera Setup

File Location

`Animate\Cameras\Cameras.tbk`

Animator: Groups

The **Animator: Groups** screen is used to group shapes, reference frames, and other groups. Due to the versatility of the animator, this library can be used for several purposes.

Discussion

The Animator draws several kinds of objects: a flat grid for the ground, a 3D grid for the ground, a target path on the ground, multiple shapes, and multiple wheels. The multiple shapes can be fixed or moving. The grid properties are specified in the **Animator: Camera Setup** screen, the target-path display properties are specified in the **Input: Target Path For Closed-Loop Steer Control** screen, and almost everything else is grouped using this screen.

Grouping Shapes Together

Groups of shapes and wheels are grouped and associated with reference frames to build a detailed visual representation of a road or vehicle.

As explained in Chapter 6, reference frames are used to define motions of vehicle parts. As the inputs to the animator are processed, each shape or wheel is moved with a reference frame.

When the animator is reading input data, there is always a single active reference frame. By definition, each shape and wheel that is processed moves with this frame.

It is important to understand that **all objects in a reference frame must be processed together**. For example, suppose there is a reference frame with the title `sedan body`. The animator will not allow some shapes to be processed for `sedan body`, then process shapes for another reference frame, and then switch back to `sedan body`. Once the animator starts processing data for a particular reference frame, it is no longer possible to add shapes or wheels to reference frames that were previously processed.

Notes:

1. Although shapes and wheels are always associated with the active reference frame, the camera and look-point coordinates are not. However, because the camera and look-point data are contained in the library **Animator: Camera Setup**, you should not have occasion to deal with them in an **Animation: Groups** data set.
2. The reason that shapes and wheels are handled differently than camera and look-point data is that shapes and wheels can be duplicated. For example, the same wheel shape is usually linked to every moving wheel reference frame. In contrast, there is only a single camera point and a single look point.

3. In CarSimEd 4.2 and older versions, this screen would include links to wheel data from an **Animator: Wheels** screen. In newer versions of CarSimEd, the information needed to draw the tire/wheel objects is taken from the tire data sets. The **Animator: Wheels** library is no longer needed and has been eliminated.

4. This data screen is unusual in CarSimEd because it is common for one data set to have links to other data sets in the same library. This is done when more links are needed that can be specified in one data set.

An animator group can be set up several ways, but the most common one is to group several shapes or wheels together without associating them with a reference frame. For example, Figure 9.2 shows how the body of a car is assembled from many individual shapes.

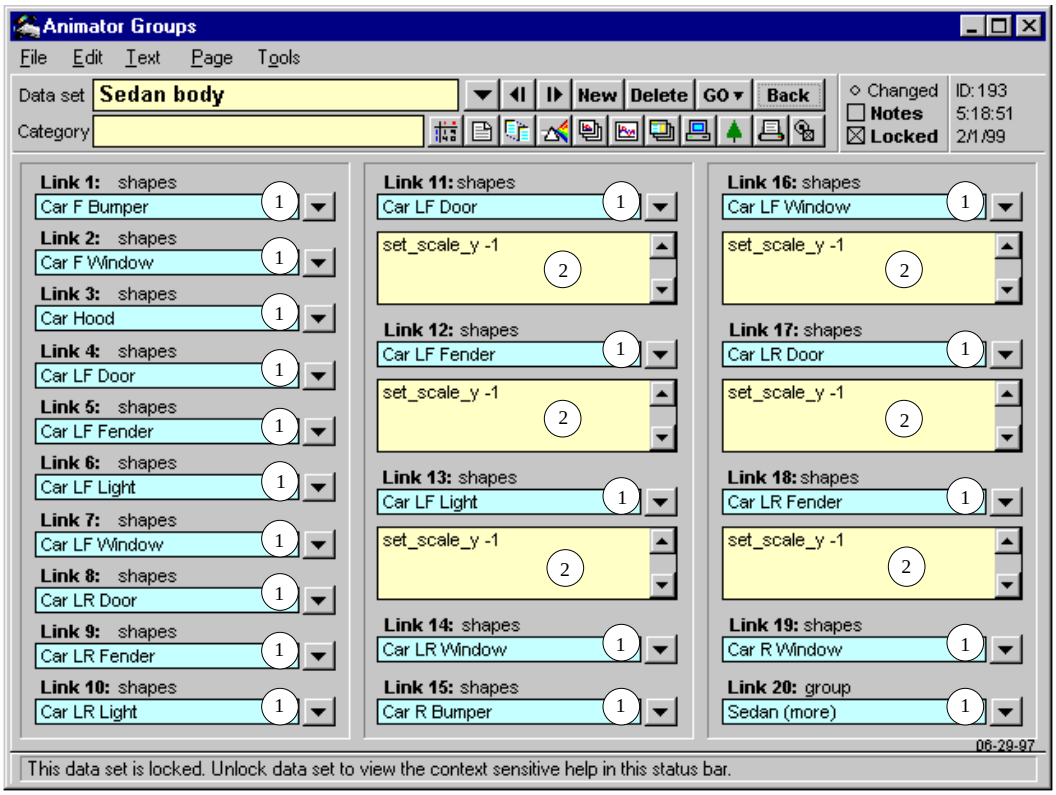


Figure 9.2. Animator group for body assembly.

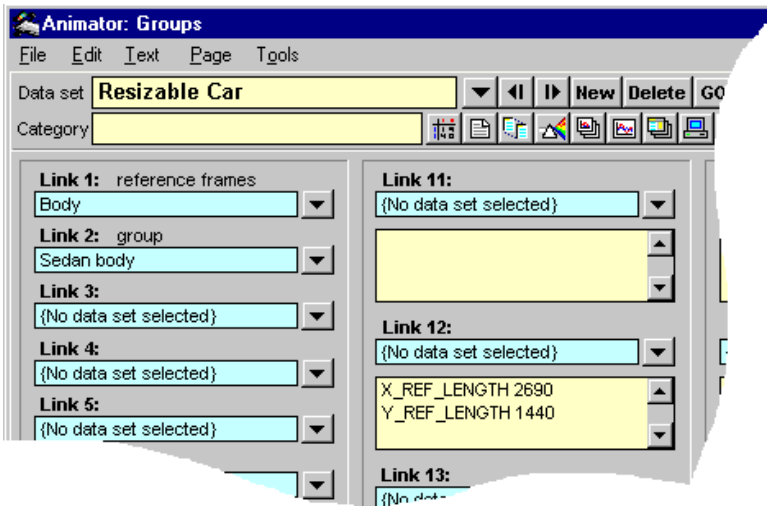
When combining reference frames and other data sets, be aware that the data are sent to the animator in the same order they are numbered on the screen: top to bottom, then left to right. If one of the links is to a reference frame, it should be the first one (Link 1). That becomes the “active reference frame” until another one is defined.

Resizable Vehicle Shapes

A recently added feature of the animator is the support of “resizable shapes.” Coordinates in shapes associated with a reference frame can be re-scaled in the X, Y, and Z directions by these ratios:

- X coordinates are multiplied by the ratio: `x_length/x_ref_length`
- Y coordinates are multiplied by the ratio: `y_length/y_ref_length`
- Z coordinates are multiplied by the ratio: `z_length/z_ref_length`

The scaling is only performed in a direction if both the length and the reference length are specified. In CarSimEd, the PAR files for the vehicle data sets specify `x_length` using the wheelbase of the vehicle. The PAR files for the suspensions specify `y_length` using the suspension track width. Therefore, you can enable the animation data to be resizable by specifying the reference wheelbase using the keyword `x_ref_length` (use mm), and the reference track width using the keyword `y_ref_length`. A portion of an **Animator: Groups** screen is shown below with reference X and Y lengths.



If the simulated vehicle has a longer wheelbase than the value specified with `x_ref_length`, the animator will stretch out the shape to fit the actual wheelbase. If the simulated wheelbase is shorter than the reference, the animator will compress the shapes.

> Note: The animator group is one of the more difficult to understand screens in CarSimEd. It may help to browse through the groups that are installed with CarSimEd, to see by example how complex systems are assembled using this screen.

User Settings

(1) Miscellaneous links. Links are used to include other groups, wheels, shapes, and reference frames. A group, wheel, or shape appearing in one of these links is attached to the reference frame link most immediately above it. If none of the links are made to a reference frame, then all wheels and shapes are attached to the most recently introduced reference frame. (That reference frame is defined in a data set above this one.)

(2) Miscellaneous fields. Use these fields to assign values to arbitrary keywords. The format is that each line has a keyword and value, separated with white space (at least a single space). These keyword values apply to the previously read data, which is found from the link immediately preceding the yellow field. These fields can be used to add to shape or wheel data. For example, to mirror an object from the left to the right side, enter the keyword `set_scale_y` followed by a value –1 (see Figure 9.2). All Y coordinates will be negated for the current shape (using data from the preceding link).

> Note: The scale factors are reset to unity and the offsets are reset to zero whenever a new shape is introduced.

These fields are also used to define scale factors that are applied to an entire group of shapes. In CarSimEd this is done to stretch vehicle shapes according to wheelbase and track width.

Location in CarSimEd

> Note: This library can be applied in different ways, which means there may be links to it from libraries other than those shown below.

CarSimEd Startup
  Runs
    Vehicles: Car
      Animator: Groups

File Location

`Animate\Groups\Groups.tbk`

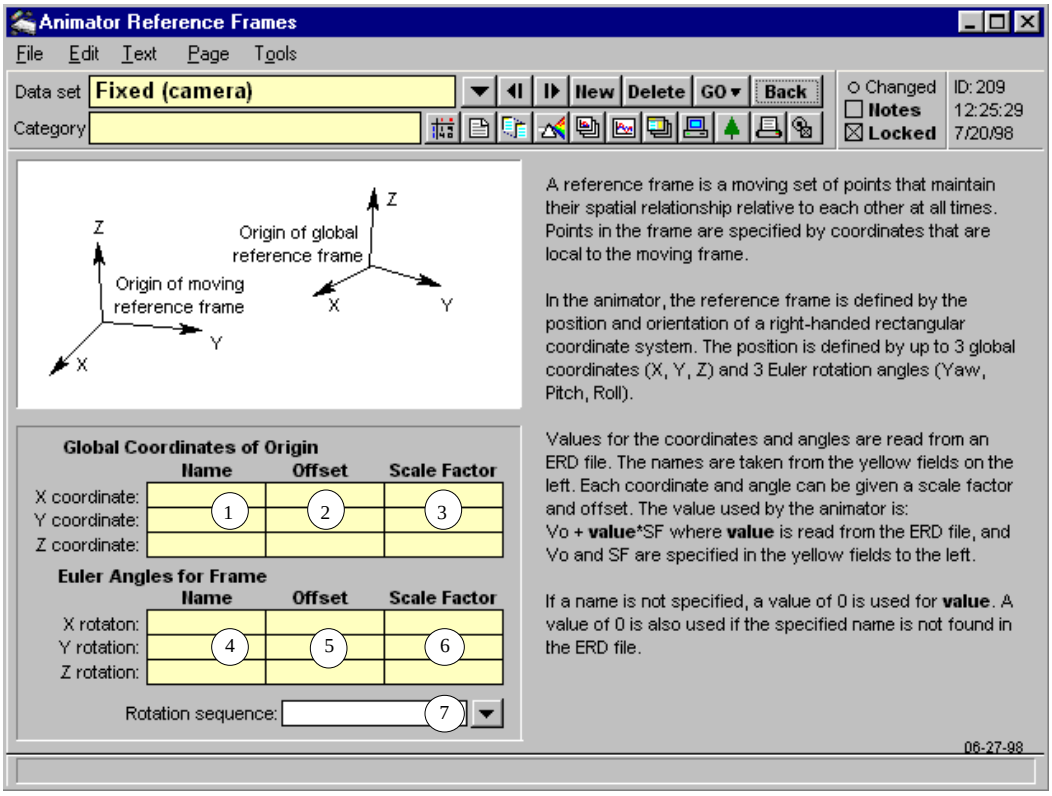
Animator: Reference Frames

This screen is used to define animator reference frames and coordinate systems (moving or stationary). In order to effectively use the animator, you need to understand the concept of a reference frame. Chapter 6 introduces the notion. This section explains how you define animator reference frames within CarSimEd.

Discussion

The animator in CarSimEd shows moving and stationary wire-frame figures. The lines used to draw each wire-frame object may move relative to other objects, but the spatial relationships between the lines in a single object are fixed.

When the relationships between a set of points does not ever change (i.e., they form a rigid structure), they are said to exist in the same reference frame. Therefore, the coordinates that are provided for shapes are constants when based on a coordinate system fixed in the appropriate reference frame.


Animator Reference Frames
File Edit Text Page Tools
Data set Fixed (camera)
New Delete GO Back
Category
Changed
Notes
Locked
ID: 209
12:25:29
7/20/98
Origin of global reference frame
Origin of moving reference frame
A reference frame is a moving set of points that maintain their spatial relationship relative to each other at all times. Points in the frame are specified by coordinates that are local to the moving frame.
In the animator, the reference frame is defined by the position and orientation of a right-handed rectangular coordinate system. The position is defined by up to 3 global coordinates (X, Y, Z) and 3 Euler rotation angles (Yaw, Pitch, Roll).
Global Coordinates of Origin
Name Offset Scale Factor
X coordinate:
Y coordinate:
Z coordinate:
Euler Angles for Frame
X rotaton:
Y rotation:
Z rotation:
Rotation sequence:
Values for the coordinates and angles are read from an ERD file. The names are taken from the yellow fields on the left. Each coordinate and angle can be given a scale factor and offset. The value used by the animator is:
Vo + value*SF where value is read from the ERD file, and Vo and SF are specified in the yellow fields to the left.
If a name is not specified, a value of 0 is used for value. A value of 0 is also used if the specified name is not found in the ERD file.
06-27-98

In order to draw the wire-frame shapes, the animator must convert relative coordinates in a moving coordinate system to absolute coordinates in the global coordinate system. This conversion is defined by the global position of the origin of the moving coordinate system (its global X, Y, and Z coordinates), along with the orientations of its three axes.

> Note: Appendix B includes a formal discussion of reference frames and coordinate systems in the context of defining output variables related to vehicle dynamics.

The orientation of a reference frame is defined by three consecutive rotations that are called Euler angles. For vehicles, the angles are commonly called yaw (rotation about the Z axis), pitch (rotation about the “new” Y axis), and roll (rotation about the “new-new” X axis).

Starting such that the X, Y, and Z axes of the moving frame are parallel with those of the global frame, the moving frame can be oriented by consecutive rotations about its axes. For example, first rotate the frame about its Z axis by a yaw angle. After this is done, the new Z direction is the same as the old, but the X and Y axes are pointed in new directions, called X' and Y'. Next, rotate about the new Y axis, Y', by a pitch angle. After the pitch rotation, the Y axis is still in the Y' direction, but the X and Z axes are pointed in new directions, called X" and Z". Finally, rotate a third time, about the most recent X axis, X", by the roll angle.

Although it might not be obvious at first if you are not experienced with 3D kinematics, any conceivable orientation of a reference frame can be described with three Euler angles. Note that the values of the angles depend on the order of the rotations: different angles are required for sequential rotations about the X-Y-Z axes than about the Z-Y-X axes.

In general, a reference frame is defined by six variables: three coordinates (X, Y, and Z), and three Euler angles. The animator reads the required six variables from the output files generated by the solver programs. After reading the six variables, each coordinate and Euler angle is calculated with a relationship of the form:

$$\text{coordinate} = C_o + C*SF_c \quad (1)$$

$$\text{angle} = A_o + A*SF_a \quad (2)$$

where C and A are the translation and angle variables obtained from the ERD file, C_o and A_o are the constant offsets, and SF_a and SF_c are scale factors (gains).

User Settings

1. Names of coordinate variables for X, Y, and Z (keywords = `set_x_name`, `set_y_name`, `set_z_name`). These are short names in the ERD files associated with the variable C in Equation 1. If no name is listed (the field is left blank), then a constant value of 0.0 is used to compute the coordinate. A value of 0.0 is also used if the name is not found in the ERD file.

2. Offsets for coordinate variables for X, Y, and Z (keywords = `set_offset_var_x`, `set_offset_var_y`, `set_offset_var_z`). These values are numbers used to replace the symbol C_o in Equation 1. If no number is entered, a value of 0.0 is used.

3. Scale factors for coordinate variables for X, Y, and Z (keywords = `set_scale_var_x`, `set_scale_var_y`, `set_scale_var_z`). These values are numbers used to replace the symbol SF_c in Equation 1. If no number is entered, a value of 1.0 is used.

4. Names of Euler angle variables for roll, pitch, and yaw (keywords = `set_roll_name`, `set_pitch_name`, `set_yaw_name`). These are short names in the ERD files, associated with the variable A in Equation 2. If no name is listed, then a constant value of 0.0 is used. A value of 0.0 is also used if the name is not found in the ERD file.

5. Offsets for Euler angle variables for roll, pitch, and yaw (keywords = `set_offset_var_roll`, `set_offset_var_pitch`, `set_offset_var_yaw`).

These values are numbers used to replace the symbol A_o in Equation 2. If no number is entered, a value of 0.0 is used.

⑥ Scale factors for Euler angle variables for roll, pitch, and yaw (keywords = `set_scale_var_roll`, `set_scale_var_pitch`, `set_scale_var_yaw`). These values are numbers used to replace the symbol SF_a in Equation 2. If no number is entered, a value of 1.0 is used.

⑦ Rotation sequence for Euler angles (keyword = `set_euler_angles`). There are 12 possible sequences of body-fixed orientation angles. A constraint is that there cannot be two consecutive rotations about the same axis. Thus, there are three choices for the first rotation, two for the second, and two for the third. However, only two sequences are common for vehicle dynamics: yaw-pitch-roll (used for large body motions), and yaw-roll-pitch, used for the spinning wheels. These are the only two allowed by the animator.

For the animator in CarSimEd, the value must be either the text `yaw_pitch_roll` or the text `yaw_roll_pitch`. These two options are selected from a pull-down menu.

Location in CarSimEd

Camera Setup

CarSimEd Startup
  Runs
    Animator: Camera Setup
      Animator: Reference Frames

Vehicle Definition

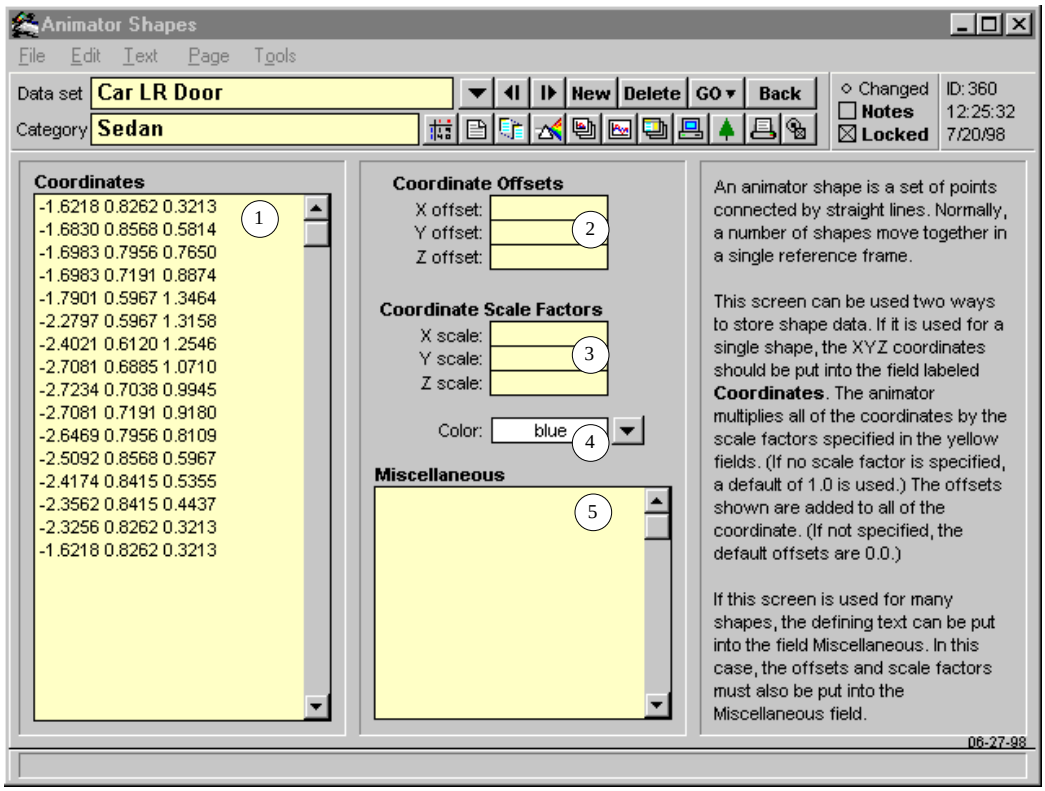
CarSimEd Startup
  Runs
    Vehicles: Car
      Animator: Groups
        Animator: Reference Frames

File Location

`Animate\Frames\Frames.tbk`

Animator: Shapes

This screen is used to define one or more wire-frame shapes that are drawn by the animator.


Animator Shapes
File Edit Text Page Tools
Data set Car LR Door
Category Sedan
New Delete GO Back
Changed
Notes
Locked
ID: 360
12:25:32
7/20/98
Coordinates
-1.6218 0.8262 0.3213
-1.6830 0.8568 0.5814
-1.6983 0.7956 0.7650
-1.6983 0.7191 0.8874
-1.7901 0.5967 1.3464
-2.2797 0.5967 1.3158
-2.4021 0.6120 1.2546
-2.7081 0.6885 1.0710
-2.7234 0.7038 0.9945
-2.7081 0.7191 0.9180
-2.6469 0.7956 0.8109
-2.5092 0.8568 0.5967
-2.4174 0.8415 0.5355
-2.3562 0.8415 0.4437
-2.3256 0.8262 0.3213
-1.6218 0.8262 0.3213
1
Coordinate Offsets
X offset:
Y offset:
Z offset:
2
Coordinate Scale Factors
X scale:
Y scale:
Z scale:
3
Color: blue
4
Miscellaneous
5
An animator shape is a set of points connected by straight lines. Normally, a number of shapes move together in a single reference frame.
This screen can be used two ways to store shape data. If it is used for a single shape, the XYZ coordinates should be put into the field labeled Coordinates. The animator multiplies all of the coordinates by the scale factors specified in the yellow fields. (If no scale factor is specified, a default of 1.0 is used.) The offsets shown are added to all of the coordinate. (If not specified, the default offsets are 0.0.)
If this screen is used for many shapes, the defining text can be put into the field Miscellaneous. In this case, the offsets and scale factors must also be put into the Miscellaneous field.
06-27-98

Discussion

A shape is a set of points connected by straight lines. Each point is defined by a set of three coordinates (X-Y-Z). The animator starts with the first point, and draws connecting lines to each following point in a list.

All coordinates are assumed to be in a local coordinate system, associated with the active reference frame. (See the discussion on how shapes and reference frames are associated in the section **Animator Groups**, on page 133.)

The animator also supports offsets and scale factors. All of the coordinates are calculated using the equation:

$$\text{coordinate} = C_O + C*SF \qquad (1)$$

where C is the original coordinate, C_O is the offset, and SF is a scale factor (gain).

The offsets and scale factors allow the shapes to be relocated and resized without requiring all of the coordinates to be changed by hand. Applying an offset has the effect of relocating the shape within the reference frame. Scale factors can be used to change the size of a shape. For example, to make a hood twice as long, enter an X scale factor of 2.0.

Scale factors can also be used to mirror a shape. If the scale factor is negative, then all of the corresponding coordinates are given the opposite sign. For example, a left fender can be converted to a right fender by setting the Y scale factor to –1.

The shape library can handle two kinds of shape data:

1. A single shape is described, using the appropriately labeled fields.
2. Multiple shapes can be described, using the Miscellaneous field.

The second approach is useful if there are many shapes used to describe a complex body, and they are unlikely to be used for any other purpose. By putting the many shapes into one data set, fewer files are processed by the animator, and there is less likelihood of accidentally modifying a group.

User Settings

(1) **Coordinates** of points that are connected by lines in the animation (keywords = `set_coordinates` to indicate the start and `end_coordinates` to indicate the end of the list). Each line should have the three coordinates of a single point. The units are meters. The syntax is that each of the values is separated by at least one space. Blank lines are not allowed.

(2) **Coordinate Offsets** (keywords = `set_offset_x`, `set_offset_y`, `set_offset_z`). All coordinates in the shape are adjusted by these offsets according to Equation 1. The units are meters. If not specified, the animator uses default values of 0.0.

(3) **Coordinate Scale Factors** (keywords = `set_scale_x`, `set_scale_y`, `set_scale_z`). All coordinates in the shape are adjusted by these scale factors according to Equation 1. The scale factors are dimensionless. If not specified, the animator uses default values of 1.0.

(4) Color (keyword = `set_color`). This sets the color of the lines drawn to connect the points. Valid colors are provided in a pull-down menu.

(5) **Miscellaneous** field. This field is for advanced users. It can be used to enter multiple shapes. The format is that each line has a keyword and value, separated with white space (at least a single space). The field should look like a portion of a PARSFILE that can be processed by the animator. See Appendix E for the animator keywords and some example files.

Location in CarSimEd

CarSimEd Startup
  Runs
    Vehicles: Car
      Animator: Groups
        Animator: Shapes

File Location

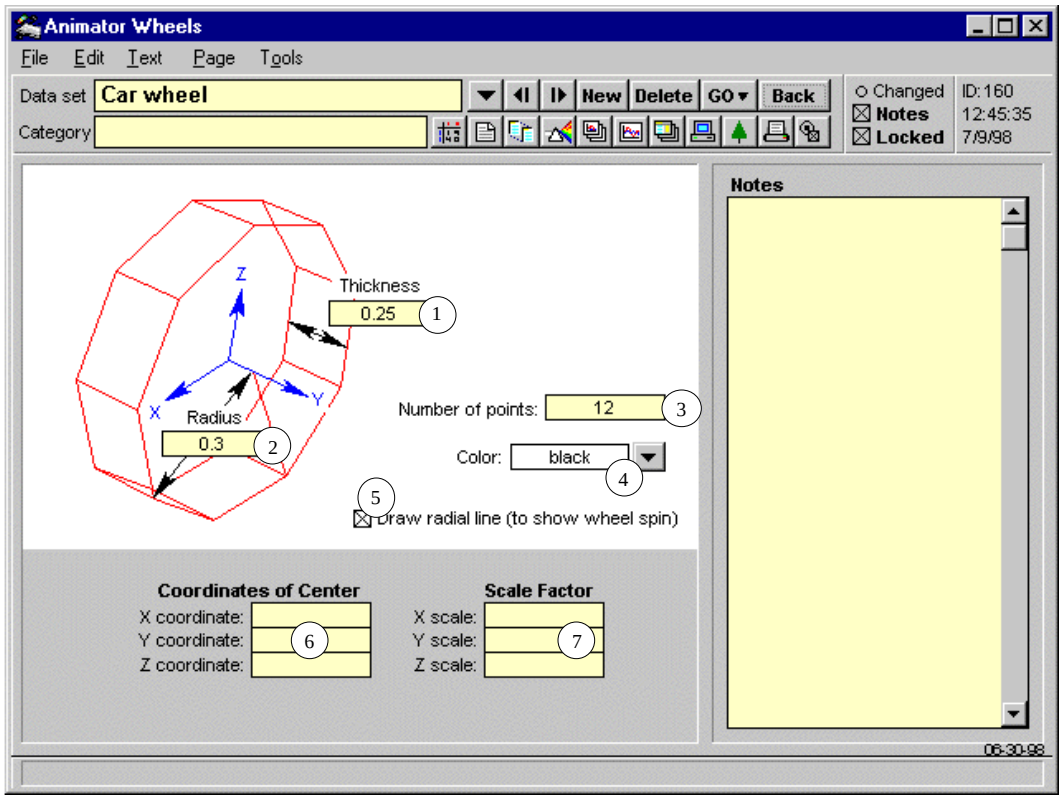
`Animate\Shapes\Shapes.tbk`

Animator: Wheels

The current version of the animator supports a single 3D graphic primitive—a cylinder. It is generally used only to define the rods and the wheel in the five-link suspension model. (Wheels on the vehicle are defined by the effective tire rolling radius from the **Tires** data screen.)

Discussion

The animator considers a wheel to be two polygons with a specified radius, separated by a specified thickness. In addition, the corresponding nodes of the polygons are connected by lines.



User Settings

1. Thickness (keyword = `set_thickness`). This should have units of meters.
2. Radius (keyword = `set_radius`). This should have units of meters.
3. Number of points used for polygon approximation of circle (keyword = `set_num_points`).
4. Color of lines drawn by animator to show the wheel (keyword = `set_color`). Valid colors are provided in a pull-down menu.

(5) Radial line option (keyword = `set_radial_line`). This check-box determines whether a line is drawn from the center of one of the polygons to the first node. The single line is useful for determining when wheels lock up during braking simulations. The values associated with the keyword (written into the text PAR file) are `on` and `off`.

(6) **Coordinates of center** (keywords = `set_offset_x`, `set_offset_y`, `set_offset_z`). These are the X, Y, and Z coordinates of the wheel center in the reference frame with which the wheel is associated. Default values of 0.0 are used if no values are provided.

(7) **Scale factors** in three directions (keywords = `set_scale_x`, `set_scale_y`, `set_scale_z`). These are dimensionless scale factors in the X, Y, and Z directions. They are multiplied by the coordinates of the wheel, centered at 0,0,0, before the offsets (6) are added. The Y scale factor adjusts the thickness, and the other two can be used to resize the wheel. If the X and Z scale factors are not equal, the wheel will be elliptical, rather than circular. Default values of 1.0 are used if no values are provided.

Location in CarSimEd

CarSimEd Startup
  Runs: Suspension Analysis
    Suspensions: 5-Link Independent
      Animator Groups
        Animator Wheels

File Location

`Animate\Wheels\Wheels.tbk`

Calculator

Use this screen to create and edit tabular numerical data. It is available from a button on the ribbon bar and from the **Tools** menu.

Discussion

This data screen is a tool supplied in CarSimEd because there are many times that simple calculations are needed when preparing numbers for CarSimEd data sets. This screen can be used to convert units for existing data, or to create tables from scratch. For example (explained in detail later), the screen display shows how a series of X-Y coordinates are created for a circular input path.

> Note: This screen does not directly feed numbers to a solver program. To use the data, you must copy the numbers to the clipboard and then paste them into the appropriate data screen.

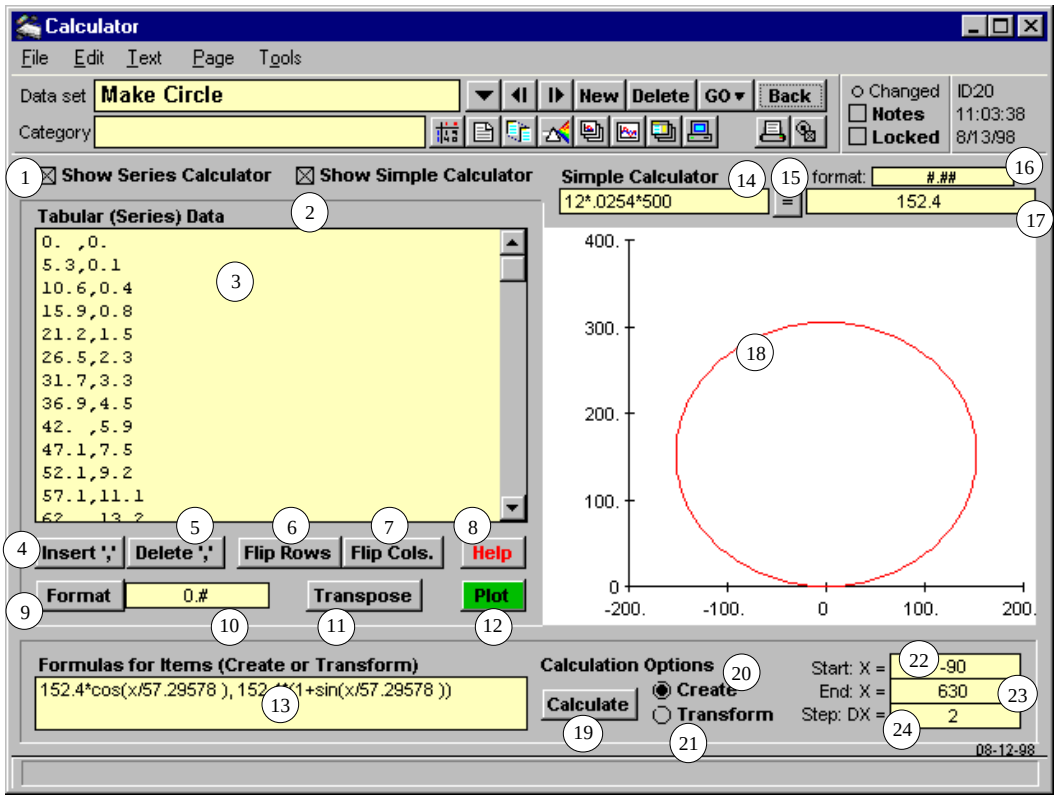


Figure 9.3. The calculator screen.

The screen design is fairly complex, offering the following capabilities:

- It plots tabular data for one or more pairs of X-Y data points.
- It performs simple calculations by evaluating a mathematical expression and printing the numerical result.
- It evaluates mathematical expressions involving values in a 2D table, updating the entire table with a single button click. For example, you can convert the units for different columns in a table using different scale factors for each column.
- It creates tabular data from scratch, based on start and end conditions for a series.

After the settings are described, a few examples are provided to show you how they are used.

User Settings and Controls

(1) **Show Series Calculator** check box. If this is not checked, most of the items on the screen are hidden. Only those involved with the simple calculator are not affected by this box ((14) - (17)).

(2) **Show Simple Calculator** check box. If this is not checked the items associated with the simple calculator ((14) - (17)) are hidden. All other items are unaffected.

(3) **Tabular Data** field. This is where the tabular data values are created and edited. The data are assumed to fit in a tabular organization, with each line in the field representing a row. Each row should have the same number of items (columns), separated by commas and/or spaces. All values in the table must be numbers.

The values in this field are shown graphically in the adjacent plot (18).

The buttons underneath ((4) - (11)) are used to change the precision of the numbers, determine whether commas separate the numbers, and to modify the column-row structure of the table.

The fields and buttons in the bottom area of the screen ((13), (19) - (24)) are used to perform calculations to create the tabular data from scratch, or to transform existing values.

The only part of the screen that is not related to the tabular data field is the simple calculator, with items (14) - (17).

(4) **Insert ‘,’** button. Click to ensure that all numbers in a row are separated with commas. If commas already exist, this has no effect except possibly to tidy the appearance of the table (3).

(5) **Delete ‘,’** button. Click to ensure that all numbers in a row are separated with white space only. If commas exist, they are removed. If numbers are already separated only by white space, this has no effect.

(6) **Flip Rows** button. Click to reverse the order of the rows of data in the tabular data field (3).

(7) **Flip Cols** button. Click to reverse the order of the columns of data in the tabular data field (3).

(8) **Help** button. Click to bring up a window with a listing of functions that can be used in formulas in the calculators.

(9) **Format** button. Click to format the numbers in the tabular data field (3) according to the format string (10).

(10) Format field. The text in this field specifies whether numbers are written with fixed decimal places or in scientific notation. It also specifies how many digits are written to the right of the decimal point. The format string has the form: X.X where the X’s represent a series of place holders:

0 — place holder for a digit.

— Same as 0 except the digit is not printed unless it is needed.

E+ or **e+** — Scientific notation with sign always used for the exponent. The number of digits in the exponent is determined by the number of place holders. If used, you must put place holders (0 or #) before and after the E+ or E-.

If the format field is blank, then numbers are written to their full precision.

Following are some example format strings and the effect they have on three example mathematical expressions.

String	45/atan(1)	atan(1)/45	5/4
none	57.2957795130823	1.74532925199433e-2	1.25
#.####	57.2958	0.0175	1.25
0.0000	57.2958	0.0175	1.2500
#.######e+###	5.729578e+1	1.745329e-2	1.25 e+0
0.000000E+000	5.729578E+001	1.745329E-002	1.250000E+000

The format field is applied when you click the **Format** button (9) or the **Calculate** button (19).

(11) **Transpose** button. Click to transpose the columns and rows in the tabular data field (1). For example:

1,2,3	Becomes	1,5
5,6,7		2,6
		3,7

(12) **Plot** button. Click this to update the graphic (18) based on the current contents of the tabular data field (3). The graphic is created by plotting values of columns 2 and higher on the Y axis, against the values of column 1 on the X axis. Thus, the number of plots is N-1 where N is the number of columns.

(13) **Formulas for items (Create or Transform)**. This field is used to create or transform numbers in the tabular data field (3). The field is used for two distinctly different purposes:

1. It provides mathematical definitions of the numbers that will be calculated and stored in the tabular data field when you click the **Calculate** button (19) and the **Create** button (20) is selected. In this case, all definitions must involve only numbers and a single variable X. X is calculated automatically and given a value for each new row based on the three fields (22), (23), and (24).

2. It provides transforms of existing numbers that will be calculated when you click the **Calculate** button (19) and the **Transform** button (21) is selected. In this case, the definitions must involve numbers and the variables X, Y, Y2, Y3, ..., where X is the first number in each line of the tabular data field, Y is the second number, Y2 is the third, and so on.

In the first case, the field is cleared and all new numbers are put into it, based on the series information ((22), (23), and (24)) and the mathematical definitions. In the second case, the tabular data field is not cleared—the existing numbers are processed and then replaced by the values calculated from the formulas.

In both cases, the tabular data field will have the same number of items as the mathematical definitions field. Initially, items (numerical results of the calculations) are separated by commas. (Use the **Delete ‘,’** button (5) to remove the commas if that is required.)

In both cases, the variable X can be included in the expressions. The meaning is not always the same. In the first case, X is an arbitrary independent variable whose values are

defined by the range and interval specified in the fields (22), (23), and (24). In the second case, X is the first number in the tabular data field (3).

The variables Y, Y2, etc. are recognized only in the second case. They should not be used when generating new numbers with the **Create** button (20) is selected.

The variables and function names used in this field are not sensitive to case: x and X refer to the same variable.

Several examples are provided in the following **Examples** subsection to indicate how you might use this field.

(14) Calculator input field. Enter a mathematical expression, then click the "=" button (15) to see the result in the calculator output field (17). In addition to normal arithmetic operations, the calculator has a number of built-in functions that are described in a later subsection and which can be displayed by clicking the **Help** button (8).

(15) Calculate button. Click to evaluate the expression in the input field (14) and print the result in the output field (17).

(16) Calculator output format. This field contains a format string used to control the round-off in the calculator output (17). It is applied when you click the "=" button (15). The syntax is the same as for the other format field (10), described previously.

(17) Calculator output field. This displays the results of the calculation performed when you click the "=" button (15). The format can be specified using the format string (16).

(18) Plot. This graphic is created by plotting values of columns 2 and higher on the Y axis, against the values of column 1 on the X axis from (3). Thus, the number of plots is N-1 where N is the number of columns. The plot is not made automatically—you must click the **Plot** button (12) to create it or update it after modifying the tabular data (3).

(19) **Calculate** button. Click to replace the contents of the tabular data field (3) with calculated numbers. As noted in the description of the formulas , the operation depends on whether the **Create** or **Transform** button ((20), (21)) is selected.

When the **Create** button is selected, the number of rows created will be equal to (End – Start +1)/Step, where the values of Start, End, and Step are obtained from the fields (22), (23), and (24).

When the **Transform** button is selected, the number of rows is the same in the table field is the same after the calculations are made.

(20) **Create** button. When selected, the fields (22), (23), and (24) are displayed and the function of the **Calculate** button (19) is defined to create a new series using the definitions in the formula field (13).

(21) **Transform** button. When selected, the fields (22), (23), and (24) are hidden and the function of the **Calculate** button (19) is defined to transform an existing series using the definitions in the formula field (13).

(22) Start value. This is the value assigned to X for the first row of numbers created in the tabular data field (3) when you click the **Calculate** button (19).

(23) End value. This is the value assigned to X for the last row of numbers created in the tabular data field (3) when you click the **Calculate** button (19).

(24) Step. This is the interval used to calculate X for every row between the first and last in the tabular data field (3) when you click the **Calculate** button (19).

Examples

Generate a Circle

A circle can be approximated by a series of X-Y coordinate pairs, where X = R*cos(A), Y = R*sin(A), R is the radius of the circle, and A is an angle. To generate a set of values that define a circle with a radius of 152.4 m (500 ft) and goes through the point X=0, Y=0, and makes two passes, do the following (see Figure 9.3):

1. Select the **Create** radio button (20).
2. Set Start X (22) to –90 (the starting angle will be –90°).
3. Set End X (23) to 630 (the ending angle will be +630°, and therefore the difference will be 720°—two times around the circle).
4. Set Step DX (24) to 2 (this will create a table with 361 points, with X-Y coordinates for every 2° of arc).
5. Define the two variables in the formulas field (13):
   `152.4*cos(x/57.29578), 152.4*(1 + sin(x/57.29578)`

   Note: the x in the above formulas is the independent variable used to create the series. It is not the X coordinate of the points on the circle.
6. Set the format string (10) to: `0.#`
7. Click the **Calculate** button (19).
8. Check your work by clicking the **Plot** button (18).

> Note: When making a series for the first time, it is quicker to use fewer points. For example, instead of specifying an increment of 2° for DX (23), you might try a value of 20, to cut the computation time by 90%.

Rescale the Circle

Suppose you want to increase the radius of the circle by 20%

1. Select the **Transform** radio button (21).
2. Define the transformation in the formulas field (13): `X*1.2, Y*1.2`
3. Click the **Calculate** button (19).
4. Check your work by clicking the **Plot** button (10).

> Note: The interpretation of symbols X, Y, etc. changes when you go from **Create** mode to **Transform** mode. In the first case, there is one independent variable X, defined by the values in the fields (22), (23), and (24). In the second, X is the name for the first number in each row, Y is the name for the second, Y2 for the next, and so on.

Create an Animation Shape Data Set for the Circle

The animator requires X,Y,Z values on each line separated by spaces (no commas). The values generated above for the circle are X and Y coordinates. A Z coordinate must be inserted for each line.

1. Select the **Transform** radio button (21).
2. Enter the following definition into field (13): `x,y, 0`
3. Click the **Calculate** button (19).
4. Click the **Delete ','** button (5).

Available Functions

The compute button processes standard arithmetic operations (+, -, *, /). An exponential is indicated with the carrot character: ^ (e.g., X^2). In addition, the compute button can process the following functions:

abs(<*number*>**)** Absolute value of number.

acos(<*number*>**)** Arccosine (number in radians).

asin(<*number*>**)** Arcsine (number in radians).

atan(<*number*>**)** Arctangent (number in radians).

atan2(<*number1*>,<*number2*>**)** Arctangent of <*number1*> divided by <*number2*> (numbers in radians).

average(<list of numbers>**)** Returns the sum divided by the number of items in the list.

ceiling(<number>**)** Rounds up to nearest integer.

cos(<number>**)** Cosine (number in radians).

cosh(<number>**)** Hyperbolic Cosine (number in radians).

exp(<number>**)** Exponent (2.7182818) raised to power of number.

floor(<number>**)** Rounds down to nearest integer.

hypotenuse(<length>,<length>**)** Length of hypotenuse.

ln(<number>**)** Natural Log (base e).

log(<number>,<base>**)** Log of number in base.

max(<list of numbers>**)** Highest value in list.

min(<list of numbers>**)** Lowest value in list.

round(<number>**)** Rounds to nearest integer.

sin(<angle>**)** Sine of an angle (in radians).

sinh(<angle>**)** Hyperbolic sine of an angle (in radians).

sqrt(<number>**)** Square root of a positive number.

sum(<list of numbers>**)** Sum of a list of numbers.

tan(<angle>**)** Tangent of an angle (in radians).

tanh(<angle>**)** Hyperbolic tangent of an angle (in radians).

truncate(<number>**)** Truncates to integer.

Location in CarSimEd

Accessed from the **Tools** menu or the ribbon bar with the button:

File Location

`Sgui_lib\Calc.tbk`

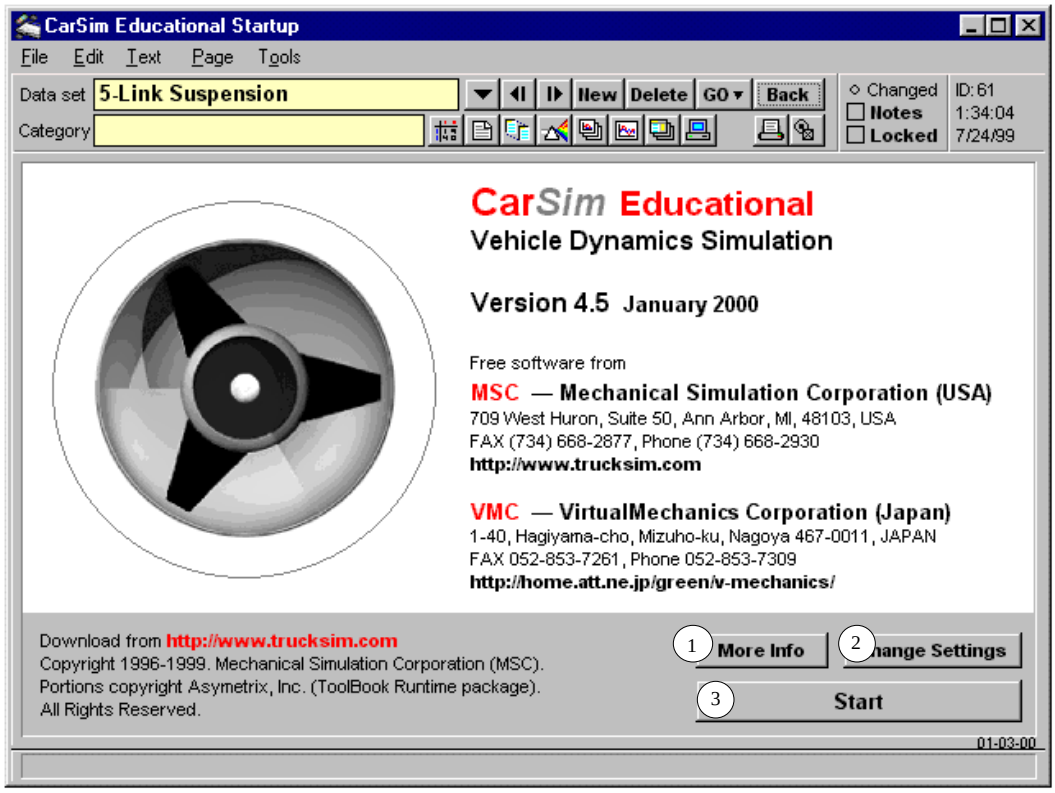
CarSimEd Startup

This screen appears immediately after CarSimEd has been started. CarSimEd has three possible entry points, and you use this screen to choose the type of simulation. After selecting the desired starting point, click the Start button to go to a **Runs** screen.

Discussion

This screen provides a starting point for the software. It also includes settings that help configure CarSimEd for your particular installation.

As installed, CarSimEd has four data sets. One, called `Install`, is required for the automatic installation to work. The other three provide entry points to the three kinds of simulation available in CarSimEd: 2D ride, 3D handling, and suspension analysis.



> Note: The **CarSimEd Startup** Screen has two possible appearances. The above figure shows the simple display.

There are at least three reasons why you might someday want to modify the CarSimEd settings or make a new Startup data set with different settings:

1. If you move the CarSimEd software to a new hard disk volume or folder, all of the text files used to communicate between CarSimEd data screens and the CarSimEd solver programs must be updated. Otherwise the solver programs will give error messages about files not being found. (The settings are made automatically by clicking the **Update** button (13) shown in Figure 9.4.)

2. If you make many data sets in a library (e.g., a **Runs** library), you may want to create a new folder and create a second library. If you add any folders to CarSimEd, you should install them in the **Startup** library.

3. You may want to change the GO menu to show fewer libraries or re-order them.

On the other hand, you may never need to change the settings contained in this library.

User Settings (Simple Display)

(1) **More Info** button. Click this button to go to screens with information about CarSimEd that involve the version, copyright information, etc.

(2) **Change Settings** button. Click this button to show additional settings that can be modified. When clicked, the CarSimEd pictures are replaced with the settings shown in Figure 9.4.

(3) **Start** button. Click this button to go to a **Runs** library. This is the most common way to start using the software. The default is that clicking the button will take you to the most recently used data set in the **Runs** library. However, you can modify the function of this button as described below.

Each data set in the **Startup** library can point to a different **Runs** library. As installed, the four installed data sets point to the three **Runs** libraries in CarSimEd.

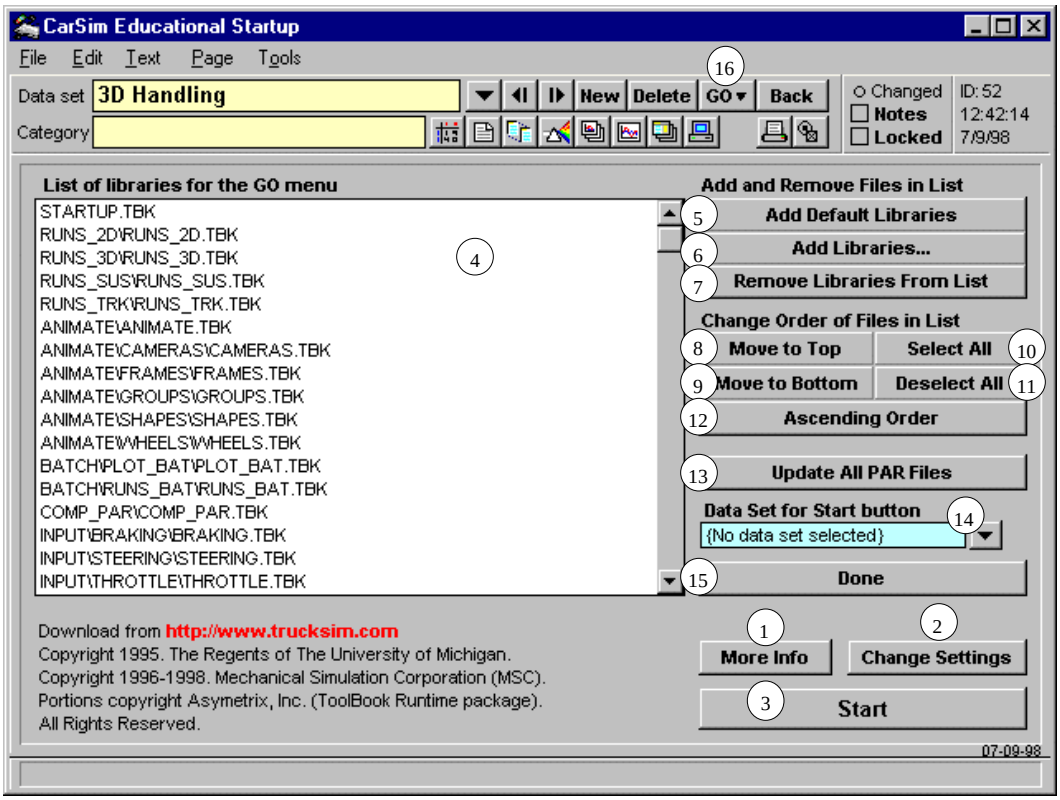


Figure 9.4. All controls of the Startup screen.

> Note: Figure 9.4 shows the **CarSimEd Startup** screen after the **Change Settings** button (2) has been clicked.

Additional User Settings and Controls

(4) List of libraries for the GO menu. This list is also used for other functions within CarSimEd — it tells CarSimEd what library files exist and where they are located.

The files are represented as pathnames relative to the CarSimEd root directory. For example, if CarSimEd is in `c:\Carsimed`, then the partial pathname

`RUNS_3D\RUNS_3D.TBK` means the full pathname for the file is `c:\Carsimed\Runs_3d\Runs_3d.tbk`.

This list is used by CarSimEd for two purposes:

1. It defines which files are affected when text data files are updated if you click the **Update All PAR Files** button (13).
2. It determines which files are listed in the pull-down menu that appears when you press the **GO** button in the ribbon bar (16).

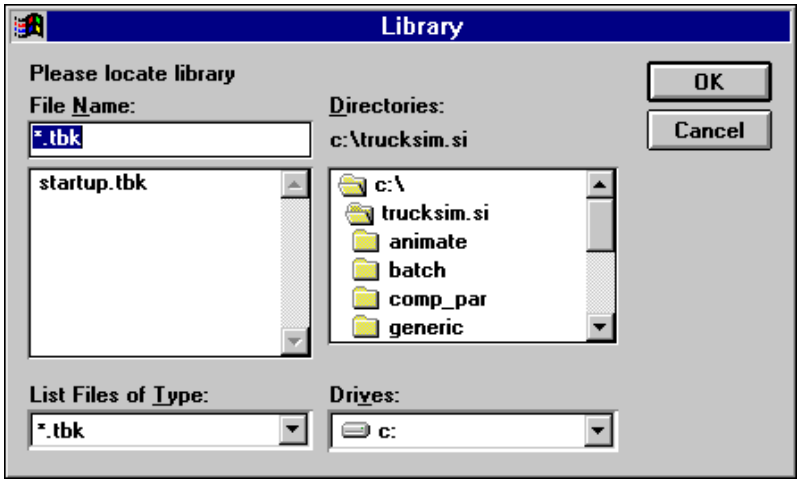
In both cases, files are included only if they are listed in the field (4) **and** if they actually exist.

This list cannot be edited directly. Lines in it can be selected by clicking on them, and the contents are manipulated by using the adjacent buttons (5) - (12).

(5) **Add Default Libraries** button. Click to add all of the library files included in the CarSimEd installation. This button is provided as a way to undo damage and return to the status "as installed." If a default library file is already in the field (4), it is not added again.

(6) **Add Libraries** button. Use this to add new libraries files to CarSimEd. For example, if you have so many data sets in a library that it is unwieldy, you can duplicate the TBK file and its folder, then delete all but one data set from the copy. Give it a new name and add it to the system.

Click the button to bring up the Windows Find File browser dialog. TBK files selected with the browser are added to the list (4).



Note: You can add more than one file at a time. After you click OK, the dialog re-appears. When you are through adding files, click the **Cancel** button.

(7) **Remove Libraries From List** button. Click to remove any selected lines from the list (4). The actual files are not affected by this—only the list is altered.

(8) **Move to Top** button. Click to move any selected lines in the list to the top. For example, to make the 3D **Runs** library the first item on the GO menu, select the line `RUNS_3D\RUNS_3D.TBK` in the list (4) and then click this button.

(9) **Move to Bottom** button. Click to move any selected lines in the list to the bottom.

(10) **Select All** button. Click to select all lines in the field (4). The contents of the field usually extend beyond the visible area, and can only be viewed by using the scroll bar. Clicking this button causes all lines to be selected, even if they are not visible.

(11) **Deselect All** button. Click to deselect all lines in the field (4).

(12) **Ascending Order** button. Click to alphabetically sort all lines in the field (4).

(13) **Update All PAR Files** button. Click to process all of the CarSimEd library files in the field (4), updating all text files. You should do this if you move CarSimEd or rename any of the TBK files or folders.

Each data set in the CarSimEd library has an associated text file with extension PAR that is used to communicate with the vehicle dynamics solver programs. These files have absolute pathnames in them. If any changes have been made to the file system (say, you moved CarSimEd from drive C to drive D), the old pathnames won't work.

The operation initiated when you click this button will probably take a couple of minutes to complete. The amount of time depends on the number of data sets in your CarSimEd installation and the speed of your computer.

(14) **Data Set for Start Button** link. This link defines what happens when you click the **Start** button (3). You can link to a library with no particular data set, or you can link to a particular data set within a library. If you do not choose a data set, then clicking the **Start** button takes you to the last data set visited in the library. This is the way CarSimEd is set when initially installed. If you would like the **Start** button to always take you to a specific data set, link to it here.

You might change this link if you move the **Runs** library, or if you create a new **Runs** library in a different folder.

What if you have several **Runs** libraries? In that case, you can make several data sets in the **CarSimEd Startup** library. Click the **New** button to make a new data set. Then, in the new data set, link to a different **Runs** library.

(15) **Done** button. Click to hide all the user settings and return to the simpler view with the CarSimEd logo.

Location in CarSimEd

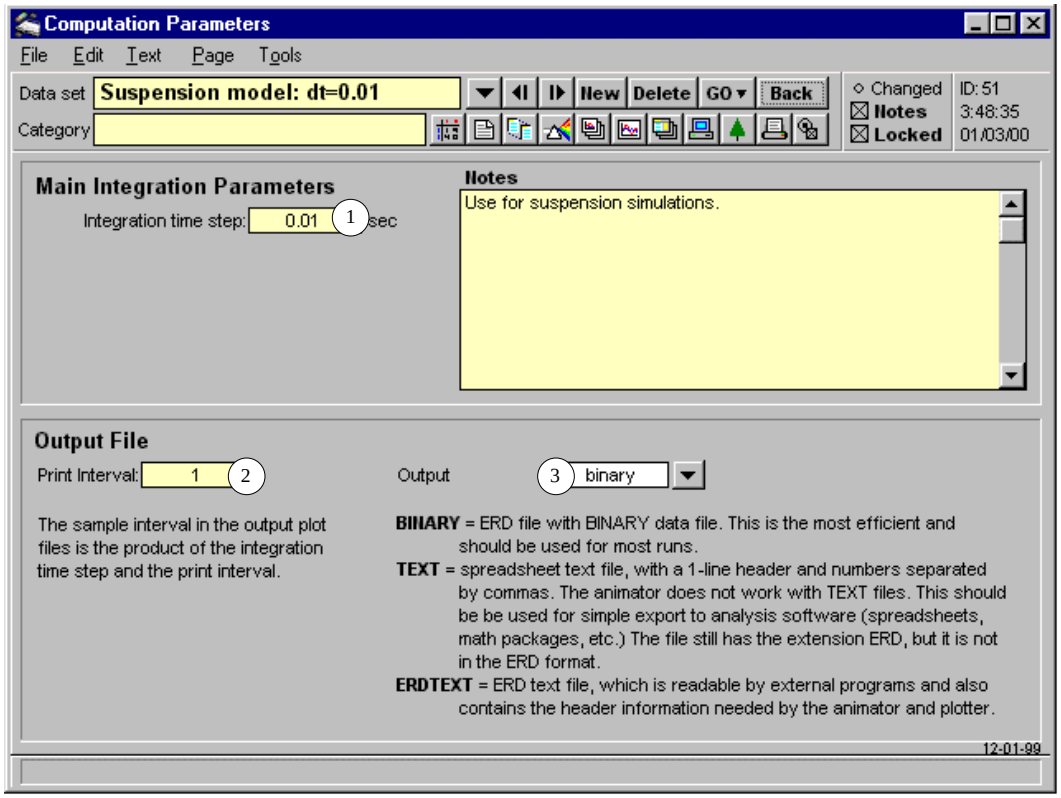
CarSimEd Startup

File Location

`Startup.tbk`

Computation Parameters

Use this screen to set parameters for the numerical integrator and to control the frequency and format of the simulation output file.


Computation Parameters
File Edit Text Page Tools
Data set
Suspension model: dt=0.01
New Delete GO Back
Category
Changed
Notes
Locked
ID: 51
3:48:35
01/03/00
Main Integration Parameters
Integration time step: 0.01 1 sec
Notes
Use for suspension simulations.
Output File
Print Interval: 1 2
Output 3 binary
The sample interval in the output plot files is the product of the integration time step and the print interval.
BINARY = ERD file with BINARY data file. This is the most efficient and should be used for most runs.
TEXT = spreadsheet text file, with a 1-line header and numbers separated by commas. The animator does not work with TEXT files. This should be be used for simple export to analysis software (spreadsheets, math packages, etc.) The file still has the extension ERD, but it is not in the ERD format.
ERDTEXT = ERD text file, which is readable by external programs and also contains the header information needed by the animator and plotter.
12-01-99

Discussion

The solver programs operate by numerically integrating a set of nonlinear differential equations over time. Taking relatively small time steps, they approximate the integration using a numerical integration algorithm.

This data screen contains parameters that have little to do with the vehicle model or its inputs. Rather, they are related to the form of the output file used to store the computed results.

User Settings

(1) Integration time step (keyword = STEP). The CarSimEd solver programs generate output files by solving equations of motion in a mathematical model of the vehicle. They repeat the calculations at small intervals of simulated time, where the interval is called a time

step. The time needed to run a simulation is inversely related to the time step. If you increase the time step by a factor of two, the program runs about twice as fast because it only makes half as many calculations. However, the calculation method is valid only for time steps that are "sufficiently small." For typical CarSimEd simulations, "sufficiently small" means about 0.002 seconds for handling runs. For braking and acceleration runs (to and from zero speed), step sizes of 0.001 or smaller might be required.

> Note: If you are making many runs with a few vehicle descriptions, you may want to determine how small the time step needs to be. Make several runs changing only the time step. For example, try using values of 0.004, 0.002, 0.001, and 0.0005 second. If the runs are valid, plots of the same variables should overlay perfectly. If the results at a large time step differ from those made with a small time step, the typical assumption is that the results made with the large time step are in error.

(2) Number of time steps between output printing (keyword = `IPRINT`). The time step (1) determines how often calculations are made. The time interval for the output files is the product of the step and this interval (IPRINT). A print interval between 10 and 30 typically works well.

> Notes: This parameter determines the resolution for plots and also the default speed for the animator. To run in slow motion (say, to accurately see wheel rotations) use a small value.
>
> This parameter has only a minor effect on the speed of a simulation run. However, it has a direct effect on the size of the output files and the time needed to load them into the plotter or animator. Big files take longer!

(3) Output file format (keyword = `FORMAT`). Use the pull-down menu to choose among the three options:

BINARY — the solver program creates a binary output file (extension = BIN) and an ERD header file. This option is the most efficient for using the integrated plotter and animator. Less time is needed to run the simulation and view output, and less disk space is needed to store information. However, binary files are not imported as easily into other software, and the contents cannot be viewed or printed with text editors.

TEXT — the solver program creates a simple text output file that can be imported into other software packages such as spreadsheets (Excel™, Lotus™, etc.) and mathematical analysis programs (MATLAB™, etc.). The first line contains short names for the output variables, separated by commas. The following lines contain numbers separated by commas. Each row has all the values for a single point in time. These files cannot be viewed with the animator.

WinEP can read text files, although the plots contain less information because the file has no title, no units, etc.

> Note: When the TEXT option is enabled, the output file generated by the solver program will have the same extension—ERD. However, the file does not follow the ERD format described in Appendix C.

ERDTEXT — the solver program creates a text ERD file. It has all the labeling information required by the animator and plotter, followed by the printed values of all output variables. These files can be imported into other programs (with some editing of the header information) and they can also be viewed within CarSimEd using the standard CarSimEd controls.

> Note: Text ERD files are not recommended for routine use of CarSimEd. The solver programs take longer to write them, and the plotter and animator programs take significantly longer to read them. Also, they occupy about three times as much disk space as the binary files.

Location in CarSimEd

CarSimEd Startup
  Runs
    Computation Parameters

File Location

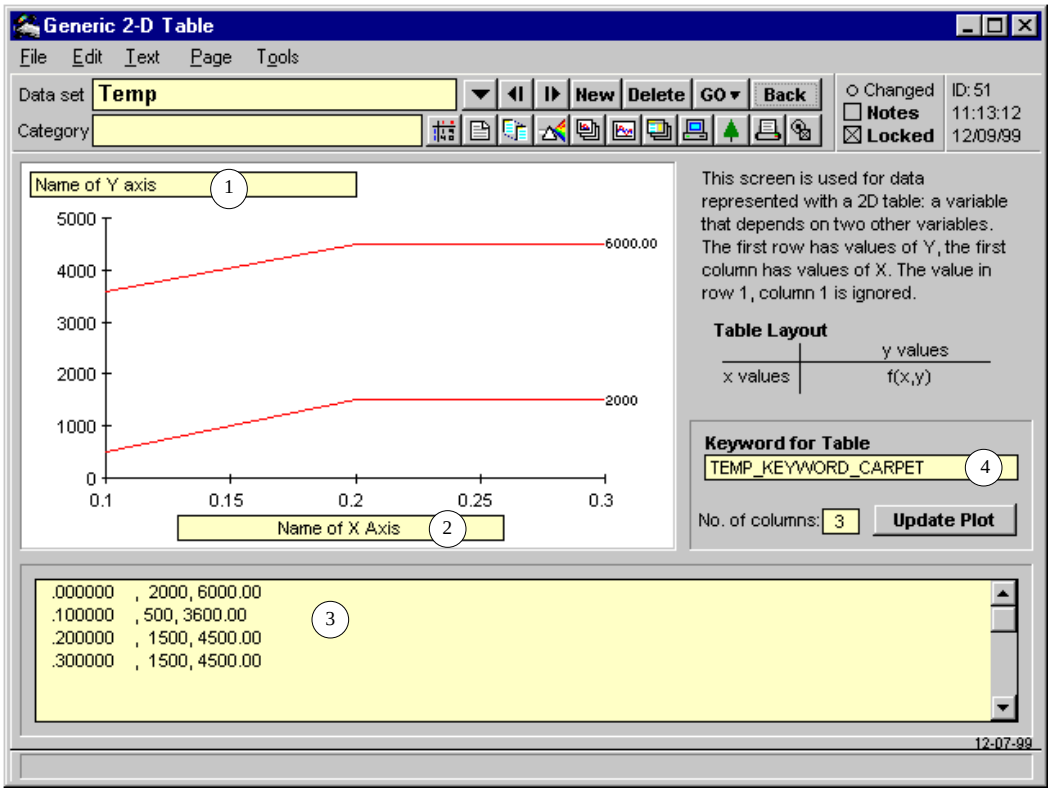
`Comp_par\Comp_par.tbk`

Generic 2D Table

Use this screen to store and display tabular data involving two independent variables for custom models. This library is not used with the standard CarSimEd models, but is provided in case you create a new solver programs with AutoSim and want to run it from the CarSimEd database.

Discussion

All tables in the standard CarSimEd models are associated with libraries in the CarSimEd data base. However, AutoSim users can create new models with the same architecture as those in CarSimEd. If these new models involve tabular data with two independent variables, then this library can be used to store the data.



Unlike most of the libraries in CarSimEd, each data set in this library can represent a different kind of data. This is possible because the keyword used by the simulation solver programs to identify tabular data is a part of the data set.

User Settings

> Note: User settings that are common for all 2D tabular data screens are described in Chapter 8, in the section **2D Tabular Data**.

(1) Label for Y axis. This label is not used by the solver programs. It is a user-comment field, provided as a means for documenting the information on the screen.

(2) Label for X axis. This label is not used by the solver programs. It is a user-comment field, provided as a means for documenting the information on the screen.

(3) Tabular data. The first row has values of the first independent variable, the first column has values of the other independent variable, and all other numbers are the values of the dependent variable.

Linear interpolation is used between rows and columns. For X and Y values outside the range of the table, linear extrapolation is used.

(4) Keyword. This keyword is required for a solver program to make use of the data in the table (3). Unlike most CarSimEd libraries, the keyword is not hidden. If the keyword in this field is not recognized by the solver program, then the data will not be used. In order

for the data to be used, the keyword must be provided and it must be spelled correctly. (However, the keywords are not case-sensitive.)

Location in CarSimEd

This library does not have a default position in CarSimEd. You can link to it from any blue field in the CarSimEd libraries.

File Location

`Generic\Gen2dtab\Gen2dtab.tbk`

Generic Data Group

Use this screen to create groups of data.

Discussion

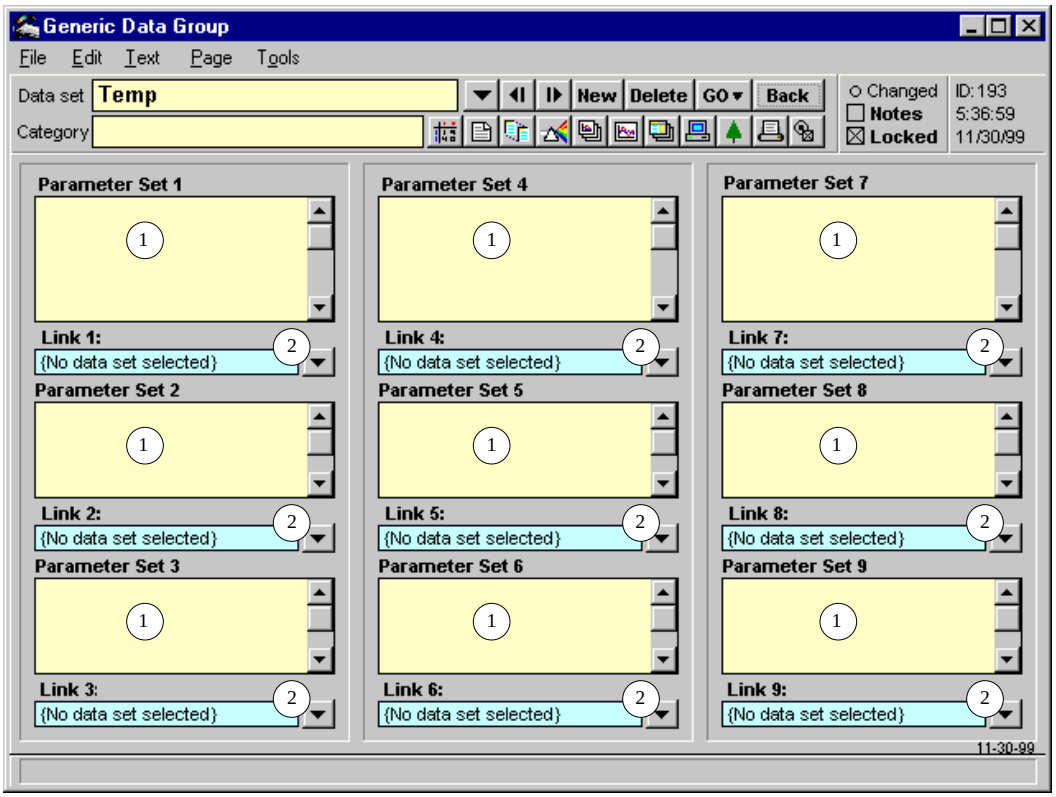
There are at least three applications for this library.

1. Create sets of related inputs, such as combinations of braking, steering, and speed.
2. Create sets of vehicle parameters that you want to apply as a group. For example, you could make a group that overrides tire data normally associated with the simulated vehicle, without making a new vehicle data set.
3. Specify parameter values that do not fit in existing data screens. For the standard CarSimEd package, there are only a few (e.g., `ROLL_STOP`, `V_STOP`). However, if you add new vehicle models, then you might want to use this library to set values for additional parameters that do not exist in the standard models.

When combining parameters and other data sets, be aware that the data are sent to the solver programs in the same order they are numbered on the screen: top to bottom, then left to right.

Many of the components are used more than once (tires, dampers, springs, etc.), and a keyword is used to determine where the description of the component should be applied. For example, the keyword `IAXLE` is used to associate axle-related data sets to axles. Put the line "`IAXLE 2`" in a yellow field, and then any links that follow will be associated with axle 2. The screen layout includes six pairs of yellow fields and blue links. Therefore, it is possible to specify data for six different places.

The best way to determine how keywords are used to locate data sets is by viewing an echo file produced by the solver program you have used (see Appendix F for an example echo file).



User Settings

(1) Miscellaneous parameter set fields. Enter keywords and the value you want assigned to them. The format is that each line has a keyword and value, separated with white space (at least a single space). The keyword values for the axle identified by the keyword `IAXLE` are used until the `IAXLE` keyword appears again in the inputs.

(2) Links are used to include other CarSimEd data sets.

> Note: You can link to other generic data groups if the number of fields and links on the screen is not sufficient.

Location in CarSimEd

This library does not have a default position in CarSimEd. You can link to it from any blue field in the CarSimEd libraries.

File Location

`Generic\Gendata\Gendata.tbk`

Generic Table

Use this screen to store and display tabular data involving a single independent variable for custom CarSimEd models. This library is not used with the standard CarSimEd models.

Discussion

All tables in the standard CarSimEd models are associated with libraries in the CarSimEd data base. However, CarSimEd can be extended to include modified or new models. If the new models involve tabular data with one independent variable, then this library can be used to store the data.

Unlike most of the libraries in CarSimEd, each data set in this library can represent a different kind of data. This is possible because the keyword used by the simulation solver programs to identify tabular data is a part of the data set.

User Settings

> Note: User settings that are common for all tabular data screens are described in Chapter 8, in the section **Tabular Data**.

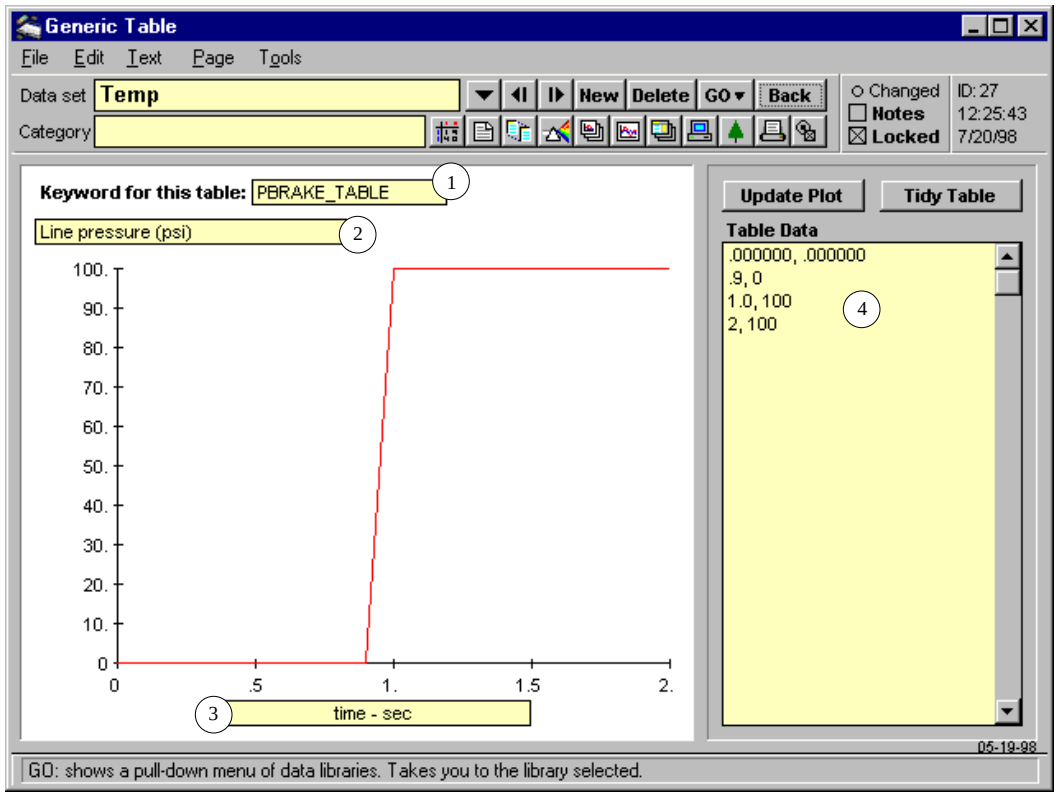


Figure 9.5. Example generic table.

(1) Keyword. This keyword is required for a solver program to make use of the data in the table (4). If the keyword in this field is not recognized by the solver program, then the data will not be used. In order for the data to be used, the keyword must be provided and it must be spelled correctly. (However, the keywords are not case-sensitive.)

(2) Label for Y axis. This label is not used by the solver programs. It is a user-comment field, provided as a means for documenting the information on the screen.

(3) Label for X axis. This label is not used by the solver programs. It is a user-comment field, provided as a means for documenting the information on the screen.

(4) Tabular data. Each row in the table should have two numbers separated by a comma. The first number is the independent variable, normally plotted on the X axis. The second is the dependent variable, normally plotted on the Y axis.

The solver programs in CarSimEd have two methods for extrapolating outside the range of a table.

1. For variables that are inputs to the vehicle (controls, such as brake pressure, or disturbances, such as wind), *flat-line extrapolation* is used. If the independent variable is less than the first value in the table, the first value of the dependent value is used. If the independent variable is greater than the last (highest) value in the table, then the last value of the dependent variable is used.
2. For variables that describe vehicle properties (for example, spring force vs. deflection), *constant-grade extrapolation* is used. Depending on whether the independent value is less than or greater than the range of the table, the first or last two points are used to extrapolate by assuming the same gradient between the dependent and independent variables.

The method of extrapolation is built into the solver program and cannot be changed. If you are in doubt about which method will be used, make sure the range of the independent variable in the table goes well beyond the range that can be covered in any simulation.

Location in CarSimEd

This library does not have a default position in CarSimEd. You can link to it from any blue field in the CarSimEd libraries.

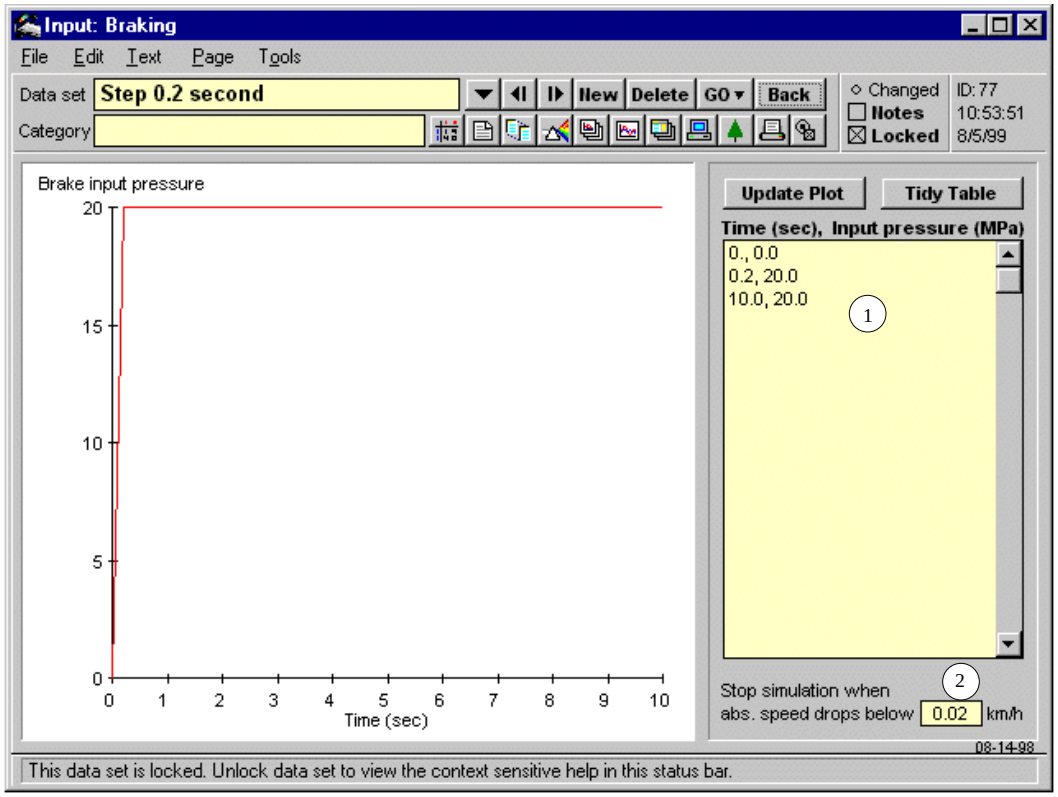
File Location

`Generic\Gentable\Gentable.tbk`

Input: Braking

Use this screen to specify the input to the brake system in terms of control pressure as a function of time.

This screen is only used for the 3D car model. If it is linked to a 2D ride or suspension analysis run, the data are ignored.



Discussion

You control braking by applying effort at the brake pedal. The brake pedal output is a control or application pressure that is applied to the brake system. This input is described via a table look-up function of brake input pressure as a function of time.

> Note: Any units can be used so long as they are compatible with the data in the **Brakes: Mechanical Properties** screen. (The units on the vertical axis of this screen must match the units on the horizontal axis in the **Brakes: Mechanical Properties** screen.) However, it's recommended that the input always be converted to units of MPa.

This screen also has a parameter that controls whether a simulation stops when the vehicle comes to rest.

User Settings

> Note: User settings that are common for all tabular data screens are described in Chapter 8, in the section **Tabular Data**.

(1) Two-column table of values of brake input pressure as a function of time (keyword = BRKIN_TABLE). Each line should have a value of time followed by a corresponding value of braking input, with a separating comma.

The solver programs use linear interpolation and flat-line extrapolation with this table. For values of time that are less than the range covered in the table, the first value of pressure is used. For values of time larger than the range covered, the last value of pressure is used. This table needs at least two lines of data or else an error message is generated.

(2) Stop speed (keyword = V_STOP). A simulation continues until one of several conditions occurs:

- The simulation time reaches the stop time specified on the **Runs** screen (keyword = STOPT).
- The vehicle roll angle exceeds a specified limit that implies that a rollover was inevitable (keyword = ROLL_STOP). (The rollover limit can be specified in the **Misc. Data** field on the **Runs** screen.)
- The absolute vehicle speed drops below a specified threshold (keyword = V_STOP).

It is the last case that is associated with braking simulations. If you want to use CarSim to determine stopping distance or other measures of braking performance, it is convenient to have the simulation stop when the absolute speed is close to zero. A typical value for this purpose would be V_STOP = 0.1 km/h.

Set V_STOP = -1 km/h To keep the simulation running after the vehicle has come to a complete stop. If the stop time continues after the vehicle comes to rest, the bouncing and other motions as the vehicle can be seen as it settles into equilibrium.

Location in CarSimEd

CarSimEd Startup
  Runs: 3D Handling
    Input: Braking

File Location

`Input\Braking\Braking.tbk`

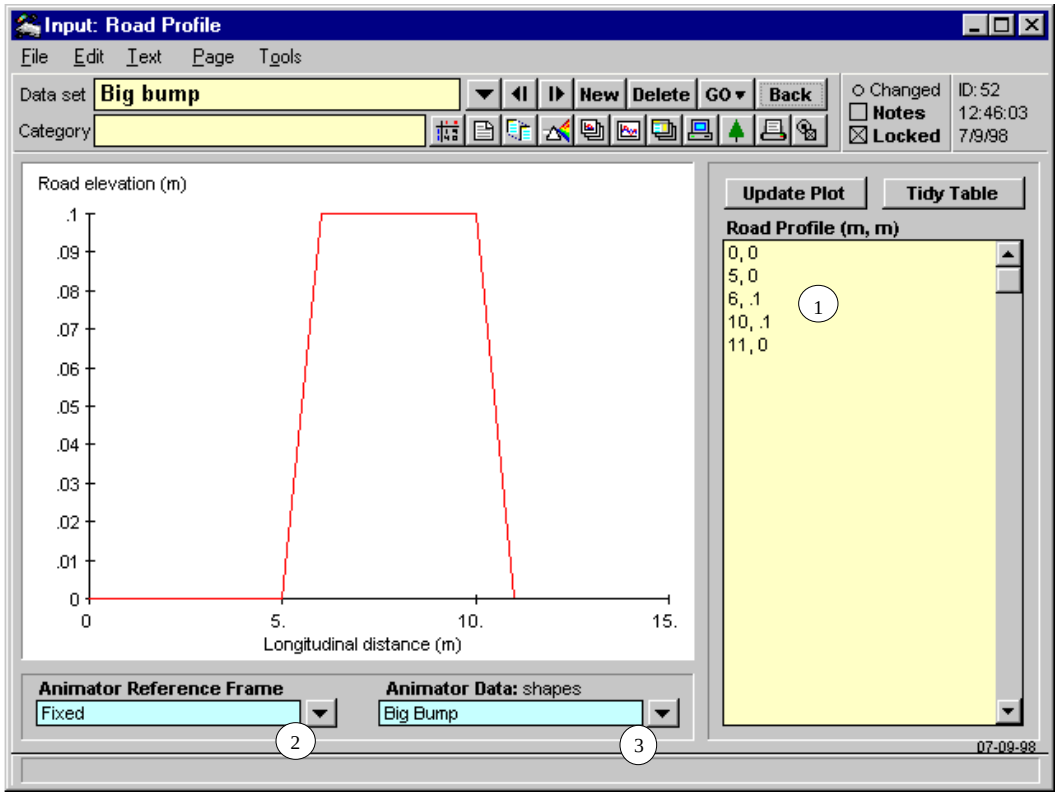
Input: Road Profile

Use this screen to specify the road input for a 2D ride run.

This screen is only used for a 2D ride run. If it is linked to a 3D car or suspension analysis run, the data are ignored.

Discussion

This data screen is used both to specify the longitudinal profile for the ride model and also to specify an optional wire-frame description for viewing the input.



User Settings

> Note: User settings that are common for all tabular data screens are described in Chapter 7, in the section **Tabular Data**.

(1) Table field for the road profile (keyword = `ROAD_PROFILE_TABLE`). Each line should have a X and Z coordinate for the road surface. The units are meters for both coordinates.

(2) Link to an animation reference frame. This is typically the installed reference frame `Fixed`.

(3) Link to an animator shape file that shows the bump or road profile.

Location in CarSimEd

CarSimEd Startup
  Runs: 2D Ride
    Input: Road Profile

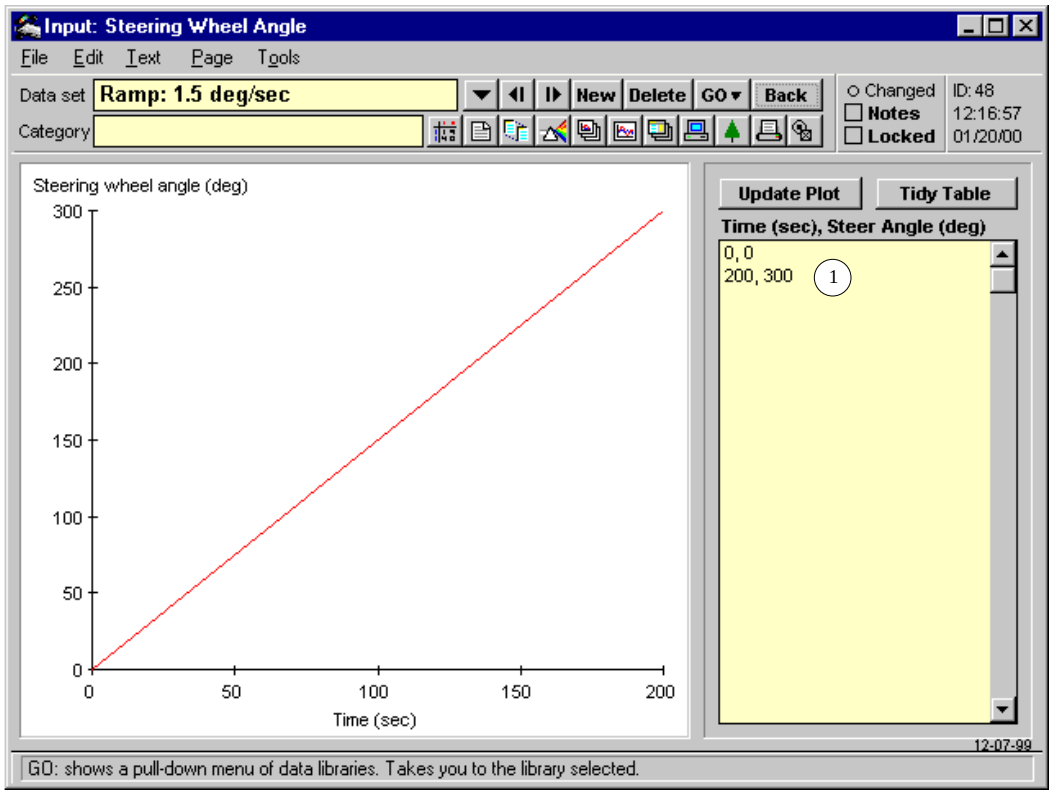
File Location

`Input\Road\Road.tbk`

Input: Steering Wheel Angle

Use this screen to define the steering wheel input as a function of time for open-loop steering maneuvers via a table look-up.

This screen is only used for the 3D car model. If it is linked to a 2D ride or suspension analysis run, the data are ignored.



Discussion

You can choose between two different ways to control steering in CarSimEd. You make the choice on the **Runs** screen by linking to a data set from one of these libraries:

1. **Input: Steering Wheel Angle** (this angle). Choose this library to apply a steering wheel angle described explicitly as a function of time, and the vehicle will be steered in an open-loop mode.
2. **Input: Target Path For Closed-Loop Steer Control.** Choose this library to apply the steering controller from the CarSimEd model.

Steering control is an input that appears as a link on the left of the **Runs** screen. To use the steering wheel input, the open-loop library must be linked as one of the inputs. The easiest way to make a new run using steering wheel input is to find an existing run made with a steering input involving the vehicle type of interest and copy that run. However, if there are no existing runs close to what you need, then you should:

1. Link to this library using one of the input links on the **Runs** screen, and
2. Make sure you are not linked to an closed-loop path following data set.

See the section **Data Links** in Chapter 8 for details on changing links.

User Settings

> Note: User settings that are common for all tabular data screens are described in Chapter 8, in the section **Tabular Data**.

(1) Two-column table of values of steering wheel angle as a function of time (keyword = STEERSW_TABLE). Each line should have a value of time followed by a corresponding value of steering wheel angle, with a separating comma.

The sign convention in CarSimEd is that positive steer involves turning to the left. (See Appendix B for details of all CarSimEd sign conventions.)

The solver programs use linear interpolation and flat-line extrapolation with this table. For values of time that are less than the range covered in the table, the first value of steering wheel angle is used. For values of time larger than the range covered, the last value of steering wheel angle is used. The table needs at least two lines of data or else an error message is generated.

Location in CarSim

CarSim Startup
  Runs
    Input: Steering Wheel Angle

File Location

`Input\Steering\Steering.tbk`

Input: Target Path For Closed-Loop Steer Control

Use this screen to define the path the vehicle is to follow in a closed-loop steering maneuver using the CarSimEd driver model.

This screen is only used for the 3D car model. If it is linked to a 2D ride or suspension analysis run, the data are ignored.

Discussion

Modes of Steer Control

You can choose between two different ways to control steering in CarSimEd. You make the choice on the **Runs** screen by linking to a data set from one of these libraries:

1. **Input: Steering Wheel Angle.** Choose this library to apply a steering wheel angle described explicitly as a function of time, and the vehicle will be steered in an open-loop mode. In this case the target path information is not used.
2. **Input: Target Path For Closed-Loop Steer Control** (this screen). Choose this library to apply the steering controller from the CarSimEd model.

Steering control is an input that appears as a link on the left of the **Runs** screen. To use the path follower model, the closed-loop library must be linked as one of the inputs. The easiest way to make a new run using the path follower is to find an existing run made with a path input involving the vehicle type of interest and copy that run. However, if there are no existing runs close to what you need, then you should:

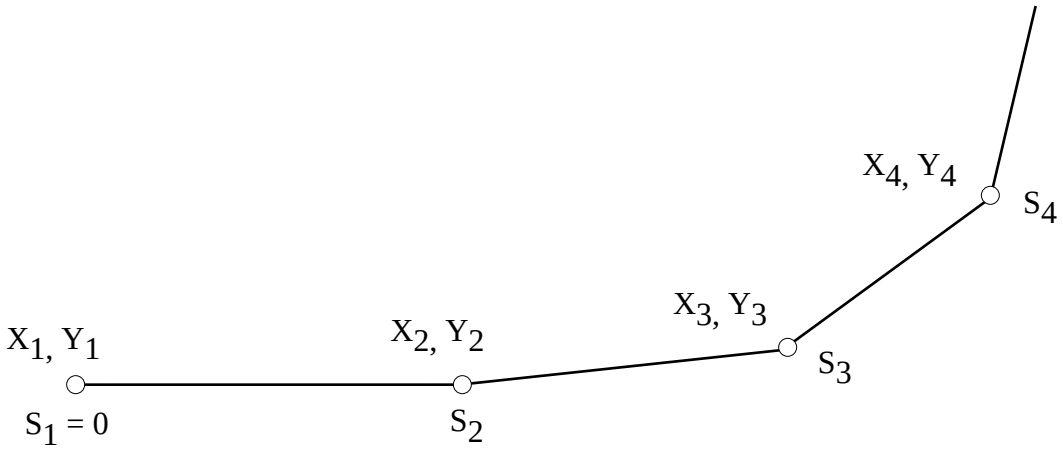
1. Link to this library using one of the input links on the **Runs** screen, and
2. Make sure you are not linked to an open-loop steering-wheel angle data set.

See the section **Data Links** in Chapter 8 for details on changing links.

Station and Path Mathematics

The table on this screen is unusual in CarSim because it does not require the rows to have an ascending order in X. Instead, The X and Y coordinates from the input path are used to compute another variable called station S, defined as the distance along the path. At the start of the path, S is defined as zero. For each pair of X-Y coordinates, a corresponding increment of S is computed by using the Pythagorean theorem. This new increment is added to the previous value of S:

$$S_1 \equiv 0 S_i = S_{i-1} + \sqrt{(X_i - X_{i-1})^2 + (Y_i - Y_{i-1})^2} \quad \{ \text{for } i > 1 \}$$

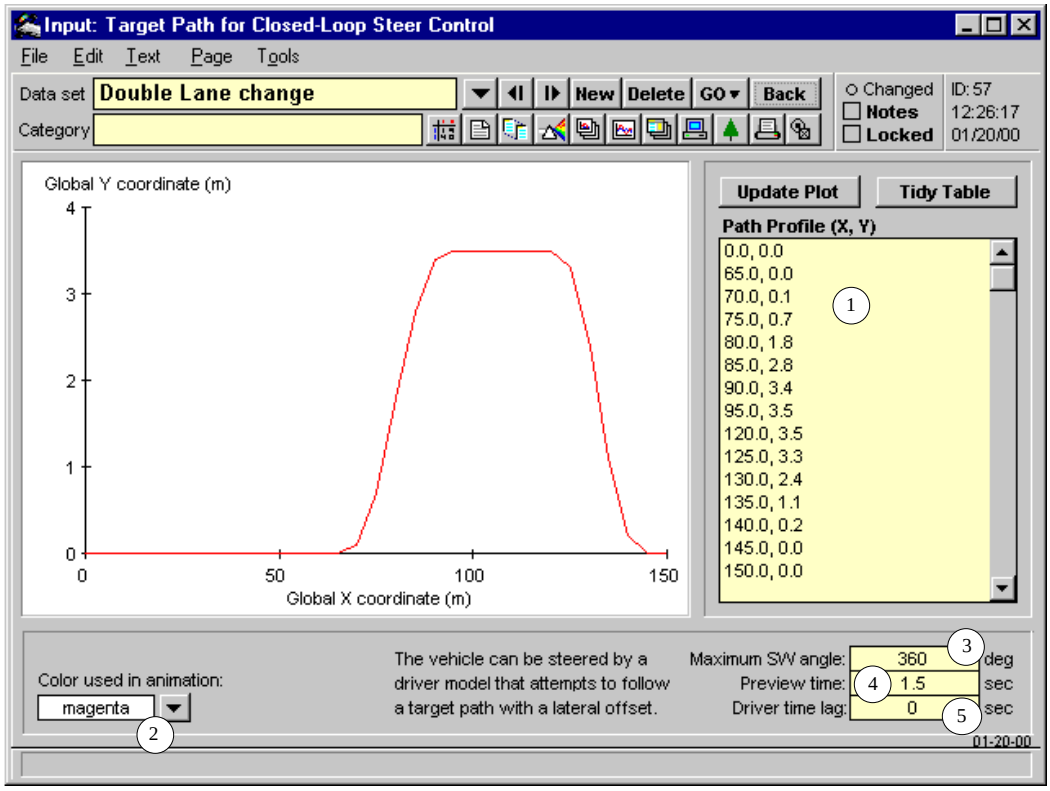


If the vehicle station is less than zero, the line connecting the first two points in the table is extended in a straight line.

If the vehicle station is higher than the largest value in the table, the last two points are extended in a straight path.

Under normal conditions, the vehicle is placed with the middle of the front axle at on the path as defined on this screen. The yaw angle is oriented so the vehicle is initially traveling parallel to the path.

The mathematics underlying the closed-loop controller are described in Appendix I.



Animation

The wire-frame animator will show the target path as a dashed line. The path is valid for the full range of station numbers covered by the vehicle in the simulation, even if it is nowhere close to the path. The animator draws the path in a color specified on the screen.

User Settings

> Note: User settings that are common for all tabular data screens are described in Chapter 8, in the section **Tabular Data**.

(1) Two-column table of values for X and Y coordinates of the target path (keyword = YIN_TABLE). Each line should have a value of X followed by a corresponding value of Y, with a separating comma. The coordinates are based on an inertial reference.

As noted above, this table is unusual in CarSim because it does not require rows to have an ascending order in X. This is possible because the table is used internally to generate two other tables: X vs. S and Y vs. S, where S is computed to ascend. Instead, the X and Y values are used to derive a third variable called station number, S.

The effect is fairly simple. The path is defined by connecting the X and Y coordinates listed in this table with straight lines (linear interpolation). For negative values of S, the first point is used to extrapolate the first gradient backward. The gradient is determined from the first two values of X and Y. For values of S larger than the range covered, the last point is used to extrapolate forward using the gradient between the last two X and Y.

(2) Animator path color (keyword = set_path_color). Use this field to specify the color of the path as drawn by the animator.

(3) Maximum SW angle (keyword = ASW_MAX). This specifies the maximum steering wheel angle that is allowed for the driver model.

(4) Preview time (keyword = TPREV). This is the look-ahead time used by the driver controller algorithm to steer the vehicle. A shorter time causes the controller to steer more rapidly in response to deviations of the vehicle from the target path. A longer time causes the vehicle to look ahead more, steering more slowly in response to changes in the target path.

A realistic value is about 1 second. For constant lateral offsets, a shorter preview time such as 0.2 seconds can be used to generate more aggressive steering to keep the vehicle on the target path. For variable paths, the short response time might lead to steering that is too aggressive and cause a loss of vehicle control.

If the table of S-L values (1) has discontinuous changes, a preview time of 1 sec is recommended to avoid over-steering by the controller.

For constant-radius turns, longer preview times can sometimes be used. The longer times produce more stable results, but lead to lateral offsets due to the curvature of the path. (To get accurate tracking with maximum stability, a lateral offset can be specified to compensate for the curvature.)

(5) Driver lag (keyword = TDLAG). Steering wheel angles generated by the driver controller are delayed by this amount of time, to simulate the neuromuscular delay in people. A realistic value is about 0.15 sec. Larger values can be used to simulated impaired drivers.

As the lag increases, the driver-vehicle systems tends to over-correct to the point of instability.

If you are attempting to follow a path closely, without trying to simulate driver response dynamics, a value of 0.0 is recommended. This results in the most accurate steering control.

Location in CarSim

CarSim Startup
  Runs
    Input: Target Path for Closed-Loop Steer Control

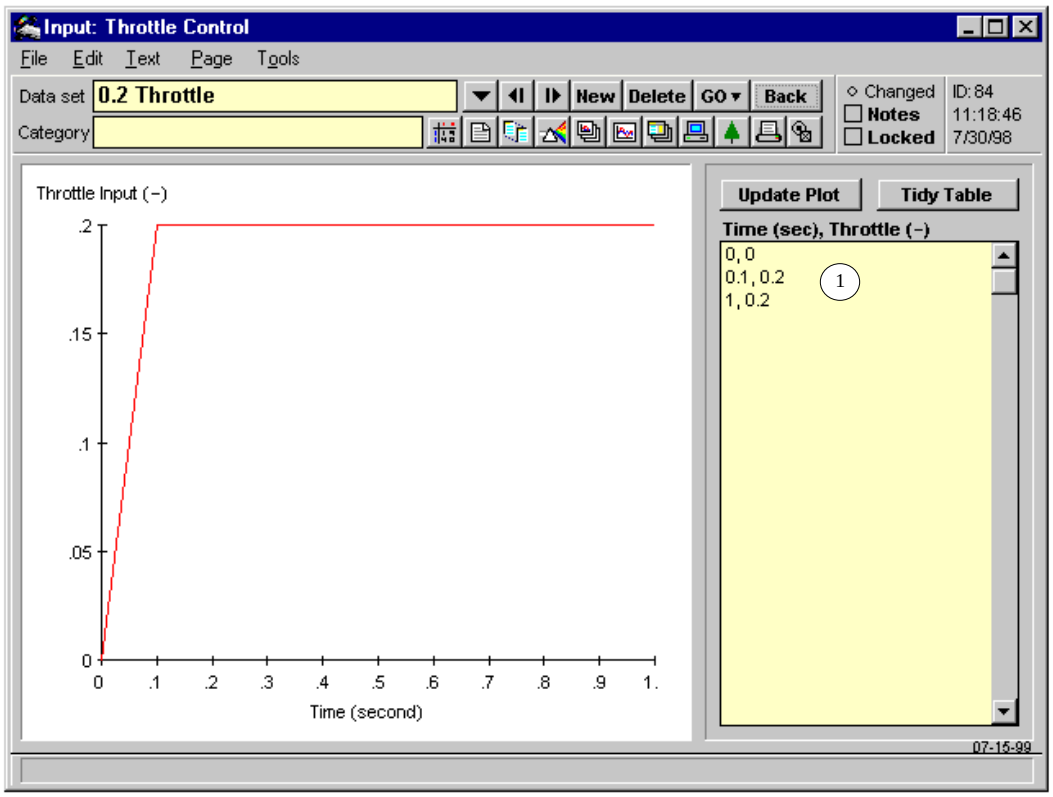
File Location

`Input\Path\Path.tbk`

Input: Throttle Control

Use this screen to define the throttle position as a function of time for a 3D handling run.

This screen is only used for the 3D car model. If it is linked to a 2D ride or suspension analysis run, the data are ignored.



Discussion

CarSimEd has two basic ways to control vehicle speed.

1. Closed-loop speed control. In this mode, you specify the vehicle target speed on the **Runs** screen and a controller is used to determine a drive torque that is

applied directly to the wheels. The assumption is that the driver "does what's necessary" to manipulate the throttle to control the speed.

2. Open-loop throttle control. In this mode drive torque on the wheels is proportional to a throttle input specifed on this screen.

The speed controller can be turned on and off using the keyword SPEED_ON_OFF. This library (throttle control) automatically sets the controller off. This means that the speed specified on the **Runs** screen is used to set the initial condition, but after the run starts, the only drive torque that is applied is proportional to the input on this screen.

If you with to run at constant speed, do not link to any data sets in this library from the **Runs** screen. That way, the speed controller will remain enabled.

Note that the simulation only runs when the absolute vehicle speed is greater than a threshold (`V_STOP`). (This allows the simulation to stop automatically in braking runs.) In order to make a run starting from a low speed, the threshold must not be greater than the starting speed. To run at any speed, including zero, set the threshold to a negative value. One way to do this is to type into the **Misc. Data** field on the **Runs** screen the line:

```
V_STOP -1
```

User Settings

> Note: User settings that are common for all tabular data screens are described in Chapter 8, in the section **Tabular Data**.

(1) Two-column table of throttle input as a function of time (keyword = `THROTTLE_TABLE`). Each line should have a value of time followed by a corresponding value of throttle, with a separating comma.

The throttle input specified with this table is multiplied by front and rear drive torque gains specified in the **Driveline** section of the **Cars** screen.

The solver programs use linear interpolation and flat-line extrapolation with this table. For values of time that are less than the range covered in the table, the first value of throttle is used. For values of time larger than the range covered, the last value of throttle is used. The table needs at least two lines of data or else an error message is generated.

Location in CarSimEd

CarSimEd Startup
  Runs: 3D Handling
    Input: Throttle

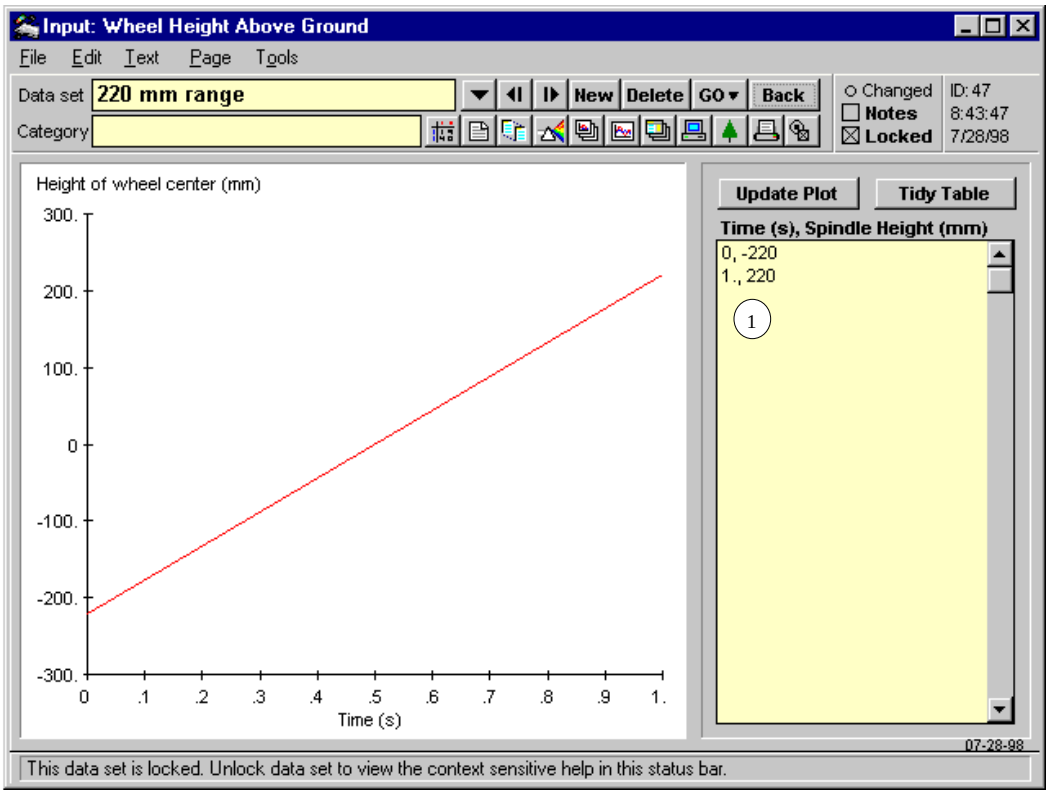
File Location

`Input\Throttle\Throttle.tbk`

Input: Wheel Height Above Ground

Use this screen to specify vertical movement of the wheel carrier (spindle) for a suspension suspension analysis run.

This screen is only used for suspension analysis runs. If it is linked to a 3D handling or 2D ride run, the data are ignored.



Discussion

The 3D suspension model is called a kinematical model because it has zero dynamical degrees of freedom. It has a single input—vertical position—that is specified as an arbitrary function of time. In this case, the amount of time covered is not important because the typical outputs of interest are cross-plots of motion variables as functions of vertical position.

User Settings

> Note: User settings that are common for all tabular data screens are described in Chapter 7, in the section **Tabular Data**.

(1) Table field for vertical wheel height (keyword = BZ_TABLE). Each line should have a value of time (seconds) followed by a corresponding height of the wheel center. Because the simulation is not dynamic, the only purpose it has in the plot is to define a continuous change in height. Two-point tables, such as the one shown above, are recommended. They simply specify the minimum and maximum ranges to be covered in the kinematical simulation.

Location in CarSimEd

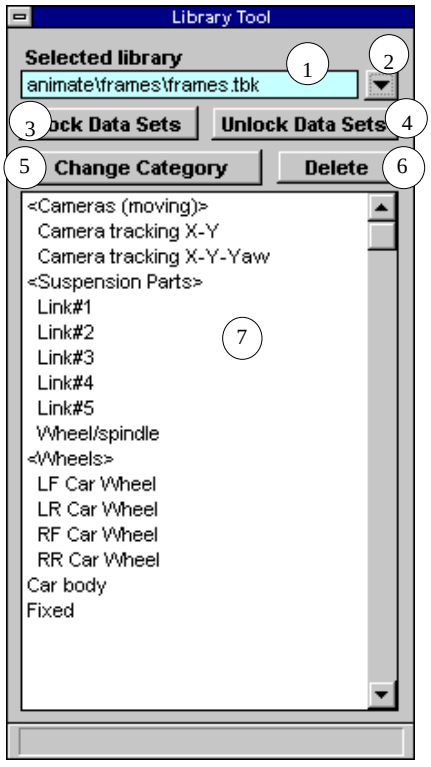
CarSimEd Startup
  Runs: Suspension Analysis
    Input: Wheel Height Above Ground

File Location

`Input\spindlez\spindlez.tbk`

Library Editor

The Library editor is a tool built into CarSimEd to help you organize libraries by performing batch delete, locking, and renaming operations.



Discussion

The Library editor is available from any CarSimEd library through the ribbon bar and the **Tools** menu. When you bring it up, it automatically links to the current library. Use it to delete a group of data sets from the library, lock and unlock multiple data sets, or to organize data sets into categories.

User Settings and Controls

(1) Link to a library. You can edit only one library at a time, and this blue field shows the one that is active. When the editor is opened, the current library is loaded into this link.

> Note: Items (1) and (2) are similar to the standard blue link. However, instead of this being a link to a data set, it is a link to the entire library. Rather than showing the name of a data set, it shows a pathname to a library file. The pathname is relative to the root CarSimEd folder. For example, if the root CarSimEd folder is `c:\CarSimEd`, then the relative pathname `Runs\Runs.tbk` refers to the file: `c:\CarSimEd\Runs\Runs.tbk`.

(2) Triangle button. Press this button to bring up a command for changing the library link shown in (1).

(3) **Lock Data Sets** button. Click this to lock all data sets currently selected in (7).

(4) **Unlock Data Sets** button. Click this to unlock all data sets currently selected in (7).

(5) **Change Category** button. Click this to change the category for all data sets currently selected in (7). It calls up a dialog box for you to enter a new category name and then modifies the category field for the selected data sets. The result is the same as if you went to each data set and changed the category field.

> Note: Categories are solely for grouping the data sets in the library and have no effect on the data set name or parameter values. This change has no effect on parameter values. It only affects how the data sets are grouped in pull-down menus.

(6) **Delete** button. This button will remove all data sets currently selected in (7). This is the same as going to each data set and clicking the **Delete** button for that screen. A warning message appears, giving you a chance to back out. **Be aware that there is no un-do option: once the data sets are deleted, they are gone!**

(7) List of data sets from the linked library. The buttons (3) through (6) affect the data sets that are highlighted. Select data sets by clicking on them. Use shift-click to select a continuous range, and use control-click to select lines that are not consecutive.

Location in CarSimEd

Accessed from the **Tools** menu or the ribbon bar with the button:

File Location

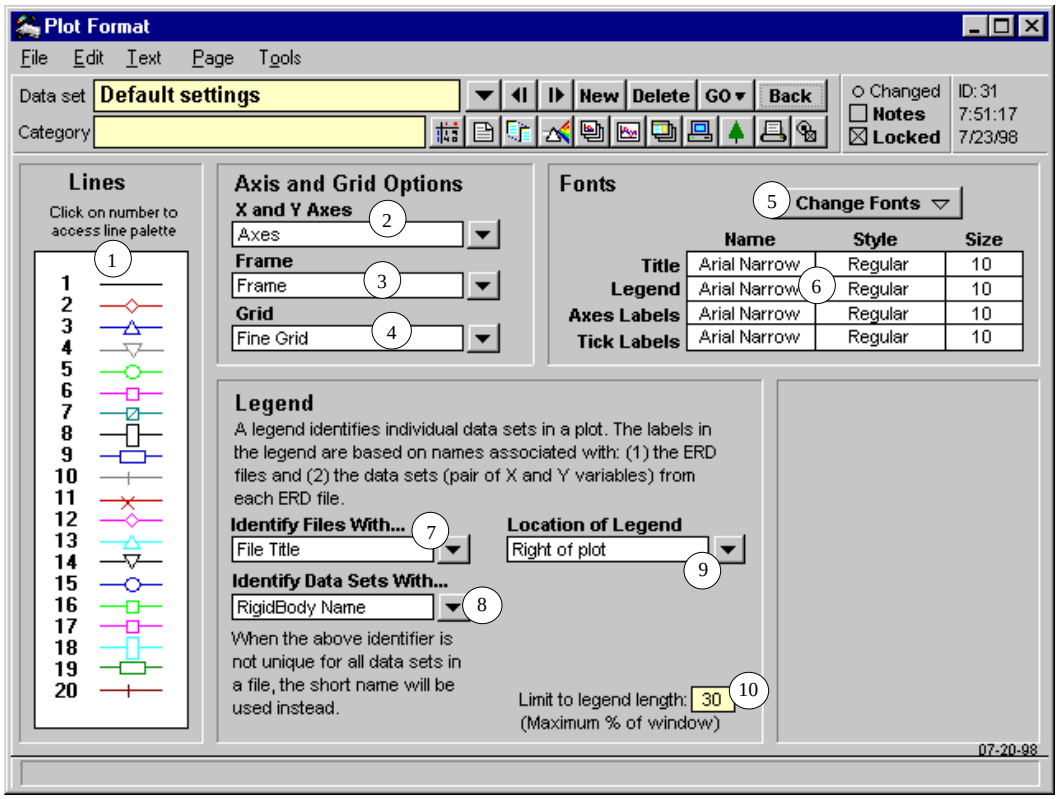
This editor is not contained in a conventional CarSimEd library file. It is stored with SGUI code in `Sgui_lib\Bootfile.tbk`.

Plot Format

The format screen determines the style and layout properties of a plot. These formats cover line style, axis and grid options, and font properties.

Discussion

The WinEP program supports a number of options for controlling the format of the generated plots. As was described in Chapter 7, dialog boxes are used for setting all options interactively, and those options can be stored in text files for future use.

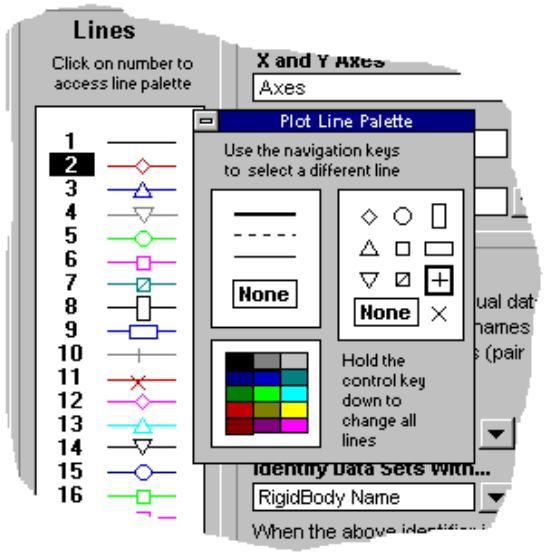


This data screen is also used to specify plotting formats. Every user setting on this screen can also be set interactively from within WinEP. However, when set here, the formatting information is stored in the CarSimEd database and can be applied easily to any plots

initiated from within CarSimEd (clicking the **Plot** button from the **Runs** screen or the **Make Plots** button on the **Plot Setup: Batch** screen).

User Settings

(1) Lines and symbol specifications (keywords = `symbols`, `linestyle`, `colors`). Clicking on a number or line will bring up the **Plot Line** palette, shown below. The palette applies to the highlighted line number. For example, line 2 is selected in the figure.



Choose the color, line style (dotted, thin, thick), and the symbol for the selected line. To change a different line, click on a different line number, or use the up and down arrow keys. To change all of the lines at once, hold down the control key when selecting the style, color or symbol.

(2) Axis selection. The button has a pull-down menu with two options: no axes or axes. The current selection is shown in the white field.

(3) Frame selection. The button has a pull-down menu with three options: no surrounding frame, a rectangular frame, or a frame with tick marks. The current selection is shown in the white field.

(4) Grid selection. The button has a pull-down menu with three options: no grid, coarse grid, or fine grid. The current selection is shown in the white field.

(5) **Change Fonts** button. Press this button to display a pull-down menu for selecting font properties for the title, legend, axes labels, and tick labels. Figure 9.6 identifies these parts of the plot. After making a pick from this screen, a standard Windows dialog box for specifying font properties appears and is used to specify the font properties.

(6) Current font settings. This table shows the current font settings for the four types of labels appearing in the plot. You cannot change the values shown directly — you must go through the **Change Fonts** button (5).

⑦ File identifier. The two options available from the pull-down menu are file title and file name. Because most ERD files generated with CarSimEd have numbers assigned by the database as their names, the title option is recommended.

⑧ Data set identifier. Variables overlaid in a plot are identified in a legend. Variables from the same file can be identified by the items provided in the pull-down menu.

⑨ Legend location. The button has a pull-down menu with five possible locations for the legend that identifies data sets in overlay plots. The choices are to the right of the plot, or on the four corners within the plot area. The current selection is shown in the white field.

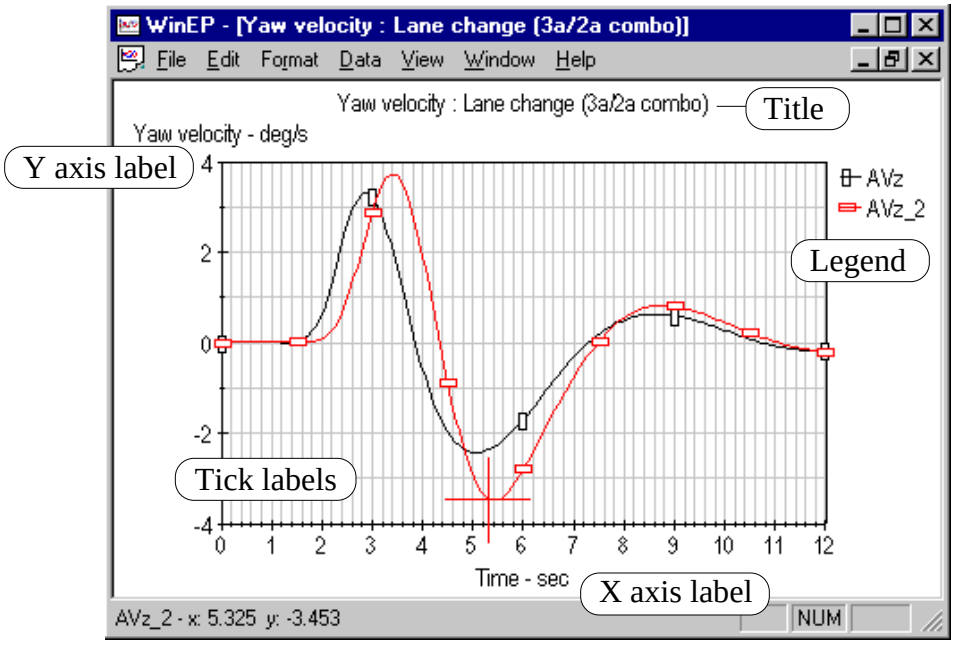


Figure 9.6. Labels in a plot.

The recommended setting is **Auto Location**, which instructs the plotter to choose one of the four locations within the plot space.

⑩ Legend size limit (keyword = `legendpercent`). When two or more data sets are overlaid, and the legend is located to the right of the plot area, WinEP sizes the plot area to leave just enough room to print the legend. In the event that at least one of the labels in the legend is long, or that the window is not very wide, then little space is available for the plot. This field is used to set a maximum percentage of the total window width that will be used for the legend.

This setting has no effect unless three conditions are met:

1. At least two data sets are overlaid in the plot.
2. The legend is placed to the right of the plot (as specified in ⑨).
3. The length of the longest label in the legend exceeds the specified percentage of the window width.

If the above three conditions are met, then the plot is sized giving the legend the specified amount. For example, if the limit is set to 30%, then the plot area is size to leave exactly 30% of the width for the legend. Labels that are too long are truncated to fit in this space.

Location in CarSimEd

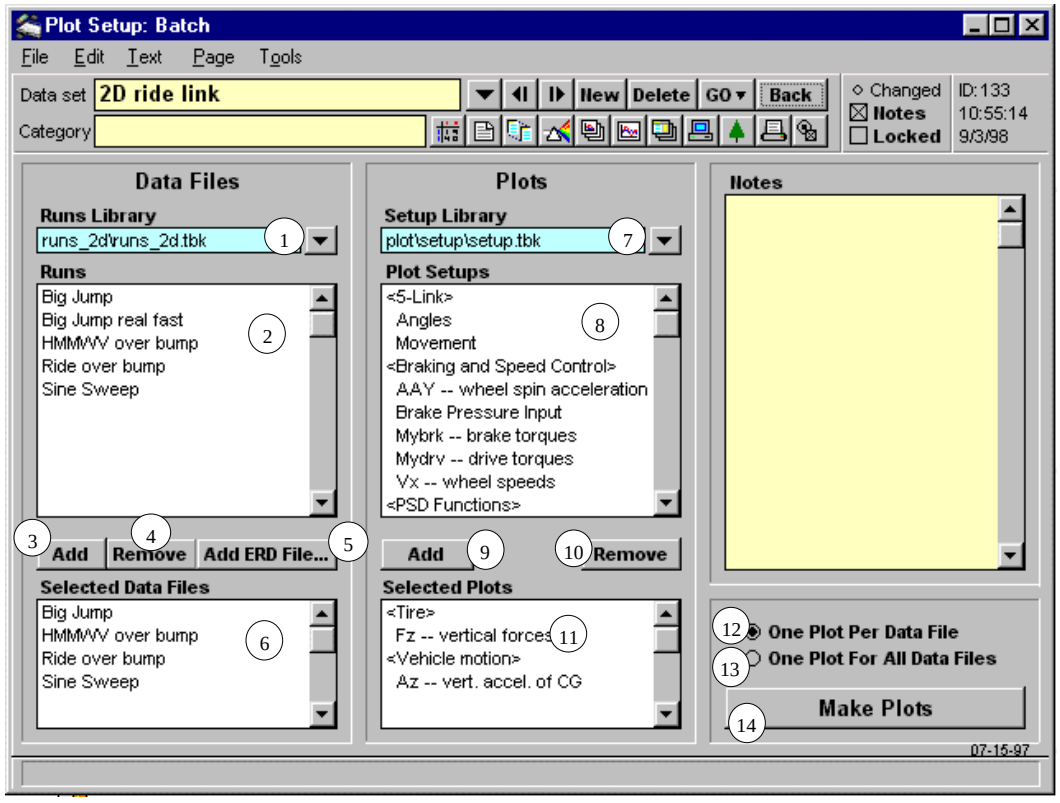
CarSimEd Startup
  Runs
    Plot Setup: Single
      Plot Format

File Location

`Plot\Format\Format.tbk`

Plot Setup: Batch

The batch plot screen is used to combine a list of runs with a list of plot conditions to automatically generate a set of plots.



Description

The standard Runs screen allows multiple plots to be specified, and the results from multiple runs to be overlaid. However, it is limited to four plot setups and three runs. If you want to trigger more plots with one button click, or overlay data from more than three runs, use this library.

User Settings and Controls

(1) **Runs Library** link. This pull-down menu has two options: (1) to go to the currently selected library, or (2) to pick a new library. The linked library must be a runs or simulation library that has corresponding ERD files. When you pick a new library, the field below is automatically updated to show the names of all of its data sets.

(2) **Runs** list. List of all of the data sets in the currently selected **Runs** library. Any line in brackets (<>) indicates a category heading. Double-click on a line to add to the **Selected Data Files** list (6).

(3) **Add** button. Click to add highlighted lines from the **Runs** list (2) to the **Selected Data Files** list (6). Shift-click to add all.

(4) **Remove** button. Click to remove highlighted data sets from the **Selected Data Files** list (6). Shift-click to remove all.

(5) **Add ERD File...** button. Calls up a dialog box for you to select ERD files that are added to the **Selected Data Files** list (6). The dialog box will continue to request selections until you click the **Cancel** button.

(6) **Selected Data Files** list. List of the selected data sets from the **Runs** list. These are the data sets that will be plotted. Any line in brackets (<>) indicates a category heading. Double-click on a line to remove from the list. This list cannot be edited directly. You use buttons (3) - (5) to modify the list.

(7) **Plot Setup** link. This pull-down menu has two options: (1) to go to the currently selected library, or (2) to pick a new library. The linked library must contain plot setup information. When you pick a new library, the field below is updated automatically to show the names of all of its data sets.

(8) **Plot Setups** list. List of all of the plot setups in the EP setup library. Any line in brackets (<>) indicates a category heading. Double-click on a line to add it to the **Selected Plots** list (11).

(9) **Add** button. Click to add highlighted lines from the **Plot Setups** list (8) to the **Selected Plots** list (11). Shift-click to add all.

(10) **Remove** button. Click to removes highlighted data sets from the **Selected Plots** list (11). Shift-click to remove all.

(11) **Selected Plots** list. List of the selected plot setups in the EP setup library. These are the plot setups that will be used for the plots. Any line in brackets (<>) indicates a category heading. Double-click on a line to remove it from the list. This list cannot be edited directly. You use buttons (9) and (10) to modify the list.

(12) **One Plot Per Data File** button. Click to cause the next set of plots to be made with a separate plot for each combination of a run and plot setup. The total number of plots will be the number of data sets multiplied by the number of selected plots.

(13) **One Plot For All Data Files** button. Click to cause the next set of plots to be made by overlaying all runs for each plot setup. The total number of plots will be the number of selected plots.

(14) **Make Plots** button. Click to start the program WinEP and instruct it to make a series of plots based upon the selected runs, plot setups, and overlay options selected.

Location in CarSimEd

Accessed with the **Tools** menu or the ribbon bar button:

File Location

`Batch\Plot_bat\Plot_bat.tbk`

Plot Setup: Single

The data sets in this library each define a graphical plot with specifications of the data (channels and files) to plot, formatting preferences, and filtering (transform) options.

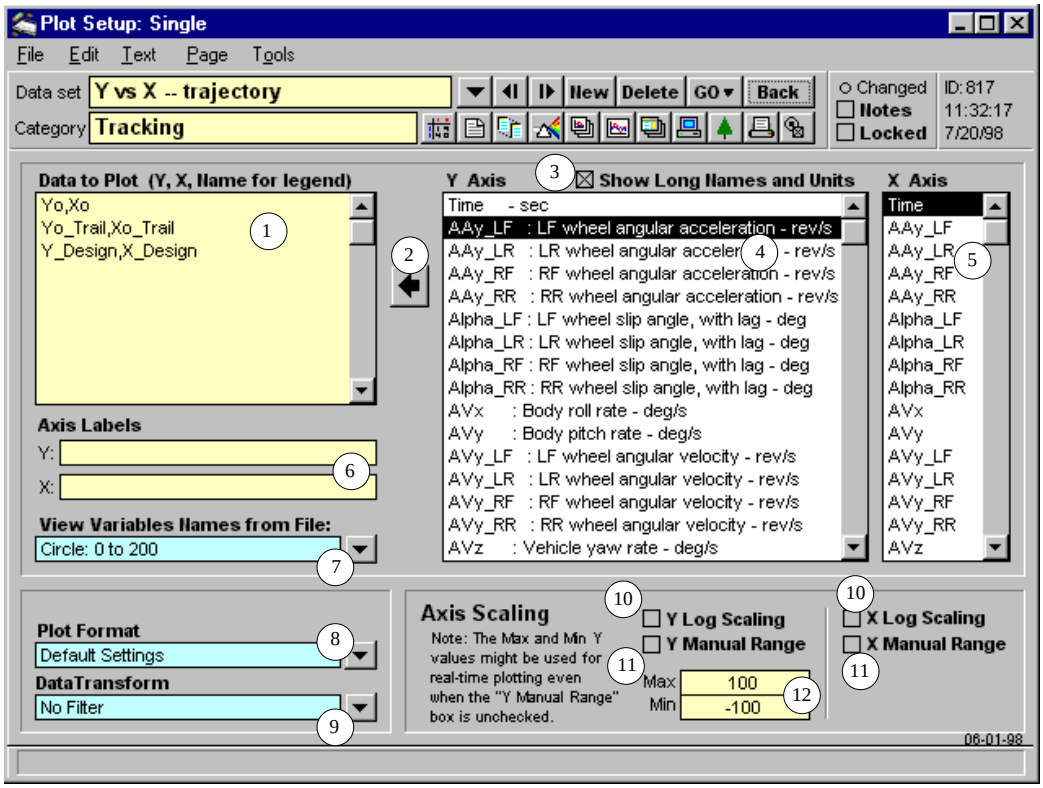


Figure 9.7. The Plot Setup screen.

Discussion

This data set is a template, used to specify what a plot should contain and how it should look. The template is applied from the **Runs** screen and the **Plot Setup: Batch** screens.

CarSimEd comes with about 30 installed plot templates. You can modify any of these, plus you can define new ones. Once a template is defined, it can be selected from the **Runs** screen and applied routinely to all runs to generate plots.

User Settings and Controls

(1) The **Data to Plot** list (keyword = `plotchannels`). This is the list of variables that will be plotted when a Plot is made. Each line contains the Y axis variable, the X axis variable, and an optional label for the data set. When the optional label is not specified, the plotter labels the data set based on standard keywords obtained from the header portion of the ERD file.

The contents of this field can be edited manually, just like any other yellow field. However, names can also be inserted with mouse clicks ((2)) by scanning an ERD file.

(2) button. Click to add the selected channels in the X and Y axis lists ((4) and (5)) to the **Data to Plot** list (1).

(3) Check box to show long names and units. When checked, CarSimEd will generate long names for display in the wide field below ((4)). On slow computers, this adds a little time to the process of scanning the file. If the ERD file contains several hundred variables, memory limits can prevent the long names from being generated.

(4) **Y Axis** list of variables. This shows all the variables in the selected run or data file (7). Double click on a line to add it and the associated line highlighted in the **X Axis** list (5) to the **Data to Plot** list (1).

(5) **X Axis** list of variables This shows all the variables in the selected run or data file (7). Double click on a line to add it and the associated line highlighted in the **Y axis** list (4) to the **Data to Plot** list (1).

(6) Optional labels for the X and Y axes. Normally, the plotter labels the axes based on keywords read from the header of the ERD files. However, you can override those labels and specify your own. If any text appears in these fields, it will be used to label the axes instead of the information from the ERD file.

(7) Link to **Runs** library or ERD file. The blue field shows the title of the currently selected run or ERD file. When a run is selected, the associated ERD output file is scanned for labels that appear in the X and Y axis fields ((4) and (5)).

(8) **Format** link. The linked data set determines the look and scale of a plot. See the section describing the **Plot Format** screen for information about the format options.

(9) **Transform** link. This is for applying a filter to the data for plotting. The available filters are moving average high, low, and band pass. See the section describing the **Plot Transform** screen for information about the transform options.

⑩ **X** and **Y Log Scaling** check boxes. When one of these boxes is checked, the corresponding axis is drawn with log scaling. Otherwise the scaling is linear.

> Note: Log scaling is only used if the box is checked **and** the associated values of the variables are all positive. If the data include zero or negative values, linear scaling is used even if the Log box is checked.

⑪ **X** and **Y Manual Scaling** check boxes. When one of these boxes is checked, the corresponding axis is scaled to cover the minimum and maximum values specified below. Otherwise scaling is performed automatically to include the full range.

⑫ Minimum and maximum values for the axes. These values are used only if the manual scaling box is checked ⑪. For the X axis, these fields are hidden when the manual scaling box is not checked. For the Y axis, they are always visible because in some versions of CarSimEd the values are needed for plotting software other than WinEP.

Location in CarSimEd

This library is accessible from several places in CarSimEd.

- Accessed from the ribbon bar with the button:
- Accessed with the **Tools** menu.
- Accessed from the Plot Setup: Batch screen.
- CarSimEd Startup
  Runs
  Plot Setup: Single

File Location

`Plot\Setup\Setup.tbk`

Plot Transforms

This screen is used for setting up plots in which the data transformed numerically. The options are to apply offsets (horizontally and vertically) and filtering (smoothing and un-smoothing).

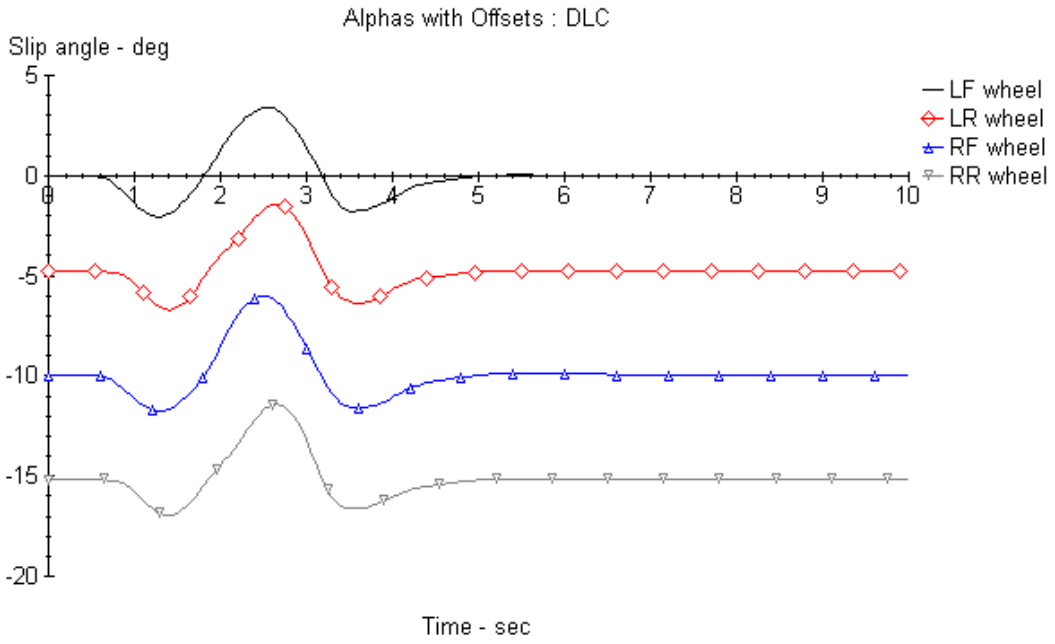
Discussion

Offsets

When comparing many similar variables, it is sometimes convenient to offset the plots. When you are viewing plots on the screen, offsetting is probably not necessary because you can see the different colors and use the movable cursor to identify the different lines.

However, if the plot will be printed in black and white, distinctions between similar traces are hard to see.

This screen has fields for specifying constants that are subtracted from the X and Y variables for each data set in the plot. For example, the figure below shows how an offset of 5 is subtracted from four plots to separate them.



Filtering

Filtering is normally used to view measured data. It is not routinely applied to simulation results generated by CarSimEd models. However, it is useful if you have occasion to view experimental data with WinEP.

WinEP includes a simple algorithm called a moving average for filtering the data. It's most basic form is for smoothing. As input it takes the original values of the variable plotted on the Y axis. Each output point is an average taken of all the adjacent points that are within a specified interval called a baselength. For example, if the baselength is 0.5, the output value at T=1.0 is the average of all values from T=0.75 to T=1.25. The output at T=1.01 is the average of all values from T=0.76 to T=1.26. This type of filter is called a Low-Pass because is filters out high frequencies while allowing low frequencies to pass through unaffected.

If the intent is to look at the high frequencies and remove the low frequencies, the same averaging is used. However, as an additional step in the processing, the smoothed values are subtracted from the originals. Figure 9.8 compares plots of data subjected to a high-pass filter and a low-pass filter to the original. Notice that the high-pass focuses on the oscillations, while the low-pass emphasizes the underlying shape of the curve.

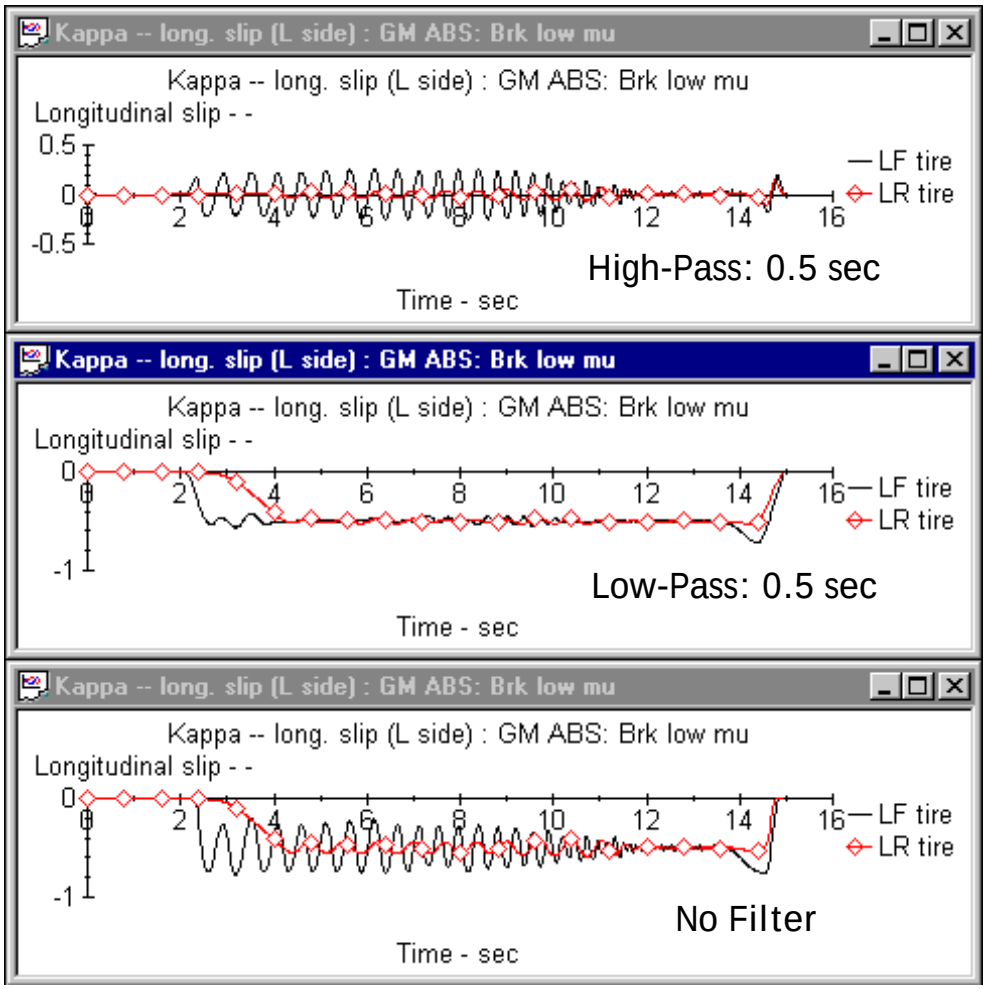
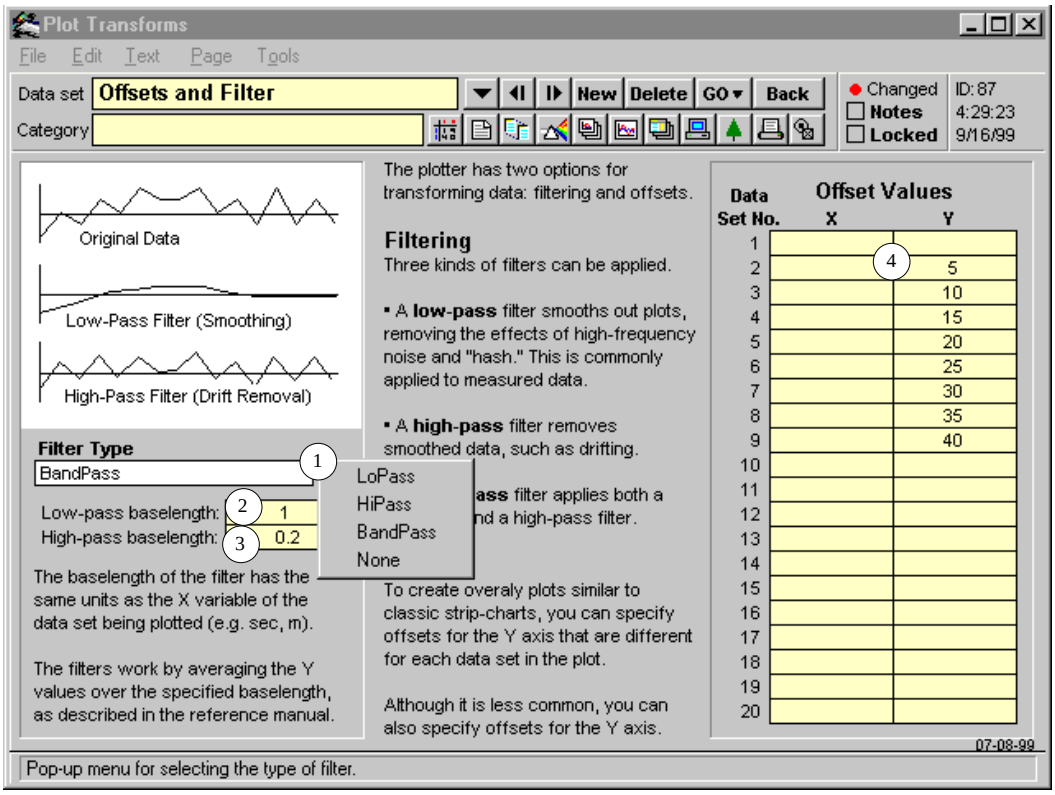


Figure 9.8. Results of filters in WinEP.

Filtering is commonly applied to experimentally measured data. High-pass filtering is used for variables that drift, such as some accelerometers. Low-pass filtering is used for examining low-frequency behavior when the measurement was subject to high-frequency vibration and possibly noise from other sources, such as accelerometers.

Sometime both high-pass and low-pass filters are applied to the same variable. This type of filtering is called band-pass.

Because simulation results usually don't have any measurement error (the exception would be if some of the inputs were taken from test results), filtering is not commonly applied to CarSimEd simulation results.


Plot Transforms
File Edit Text Page Tools
Data set Offsets and Filter
Category
New Delete GO Back
Changed
Notes
Locked
ID: 87
4:29:23
9/16/99
Original Data
Low-Pass Filter (Smoothing)
High-Pass Filter (Drift Removal)
The plotter has two options for transforming data: filtering and offsets.
Filtering
Three kinds of filters can be applied.
• A low-pass filter smooths out plots, removing the effects of high-frequency noise and "hash." This is commonly applied to measured data.
• A high-pass filter removes smoothed data, such as drifting.
ass filter applies both a
nd a high-pass filter.
To create overaly plots similar to classic strip-charts, you can specify offsets for the Y axis that are different for each data set in the plot.
Although it is less common, you can also specify offsets for the Y axis.
Filter Type
BandPass
LoPass
HiPass
BandPass
None
Low-pass baselength: 1
High-pass baselength: 0.2
The baselength of the filter has the same units as the X variable of the data set being plotted (e.g. sec, m).
The filters work by averaging the Y values over the specified baselength, as described in the reference manual.
Data Set No.
Offset Values
X
Y
07-08-99
Pop-up menu for selecting the type of filter.

User Settings

(1) Filter type. Press the adjacent button to display the pull-down menu shown in the figure, with the choices of LoPass, HiPass, BandPass, or no filter. If a filter is selected, one or both of the fields for defining baselengths are shown ((2), (3)).

(2) Low-pass baselength. This is a baselength for a moving average when the filter type is either low-pass or band-pass. This field is hidden if the filter type is no filter or high-pass.

As noted above, this defines an interval used to smooth the plot by averaging. A long baselength performs more averaging and removes more high frequencies by smoothing. A shorter baselength results in less averaging, leaving more of the original content.

(3) High-pass baselength. This is a baselength for a moving average when the filter type is either high-pass or band-pass. This field is hidden if the filter type is no filter or low-pass.

As noted above, this defines an interval used to define a smoothed set of numbers that are subtracted from the original. A long baselength removes only the static values and very low frequencies. A shorter baselength results in less averaging, meaning that more of the original data are removed.

(4) Offset values subtracted from the variables being plotted. One column has values subtracted from the variables plotted on the X axis and the other has values subtracted from the variables plotted on the Y axis. The units of the variables depend on what is being plotted.

Location in CarSimEd

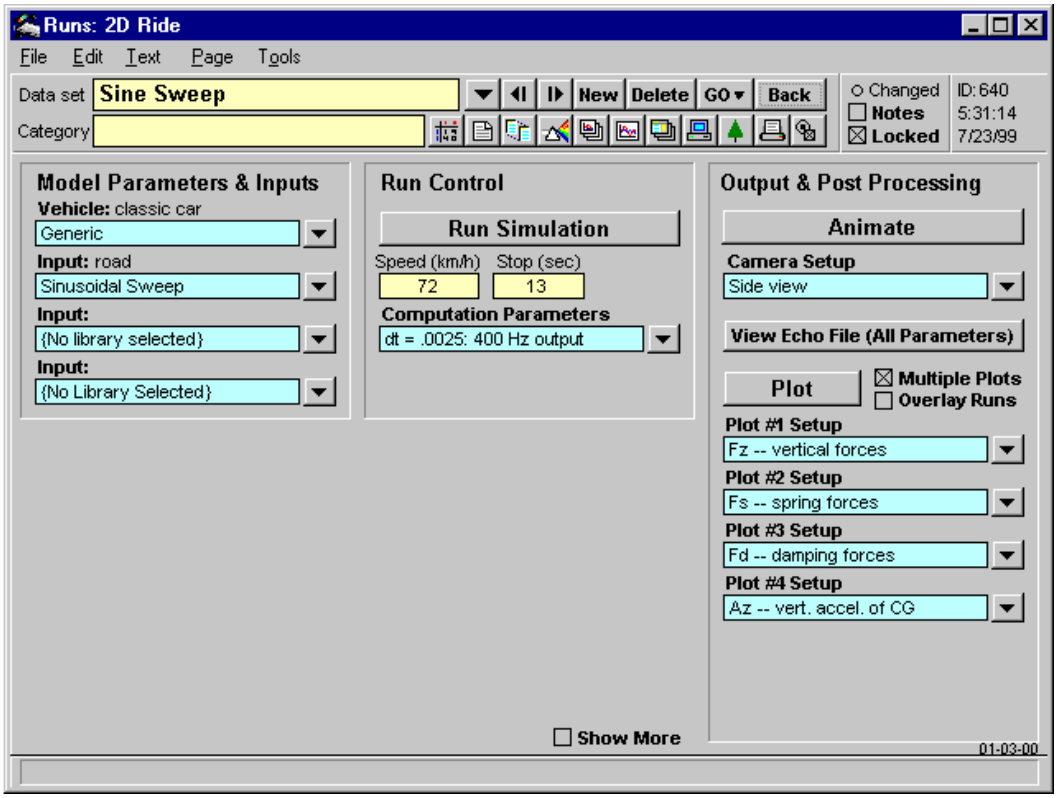
CarSimEd Startup
  Runs
    Plot Setup: Single
      Plot Transform

File Location

`Plot\Transfrm\Transfrm.tbk`

Runs: 2D Ride

The **Runs** screen is central to the user interface in CarSimEd. It is used to set up simulation runs and to view results with post-processing programs.



Discussion

CarSimEd includes four **Runs** screens. Three are for running the stand-alone solver programs (3D Car, 2D Ride, and Suspension Analysis) and the other is for running with SIMULINK. These screens are nearly identical in appearance and function. The 2D Ride screen is identical to the 3D Car screen (see Figure 9.9 on page 189), except for the title and the lack of a yellow field for specifying tire/ground friction. Please refer to the

section for the 3D Car version for descriptions of the controls and settings of this **Runs** screen.

> Note: The only meaningful input for the 2D Ride model is a road input (height Z as a function of longitudinal posistion X). Therefore, discussions of steering, braking, and throttle do not apply when using the 2D Ride model.

Location in CarSimEd

The **Runs: 2D Ride** screen can always be accessed from the **GO** menu and the CarSimEd Startup screen:

CarSimEd Startup
  Runs: 2D Ride

The **Tools** menu and the button in the ribbon bar will take you to one of the **Runs** libraries in CarSimEd. As installed, these are shortcuts for getting to the stand-alone **Runs** screen. If you primarily use the 2D ride model, you can change these shortcuts by Control-clicking the button in the ribbon bar. This brings up the file browser dialog box, which you can use to identify the file `Runs_2d\Runs_2d.tbk` as the default **Runs** library.

File Location

`Runs_2d\Runs_2d.tbk`

Runs: 3D Handling

The **Runs** screen is central to the user interface in CarSimEd. It is used to set up simulation runs and to view results with post-processing programs.

Discussion

CarSimEd includes four **Runs** screens. Three are for running the stand-alone solver programs (3D Handling, 2D Ride, and Suspension Analysis) and the other is for running with SIMULINK. These screens are nearly identical in appearance and function. This section describes the 3D version in full detail. Three other sections describe the differences between those screens and this one.

The **Runs** screen controls all aspects of a CarSimEd simulation.

Regions of the Runs Screen

Notice that the screen image is divided into three regions (see Figure 9.9).

1. **Model Parameters & Inputs** — this has links to inputs to the computer model, including the vehicle and control inputs.

2. **Run Control** — this includes parameters that control the extent of the run and a button to make the run.

3. **Output & Post Processing** — this has controls for viewing outputs generated by the run.

The **Runs** screens is special within CarSimEd because it has several buttons that cause other programs to run and do things (run a vehicle simulation, animate results, show graphs of output variables). However, like all the other screens covered in this chapter, it defines a data set within a library. In this case, the data set defines the conditions covered by a run.

Data in the left-most two regions do not have anything to do with outputs of a run. They define the inputs and parameters that will be used in the computer model if and when a run is made. In order to have an effect, changes must be made **before** a run is made. At any time: (1) any of these inputs can be changed, and (2) a new run can be made. The solver program will always get its inputs using the current data from this screen.

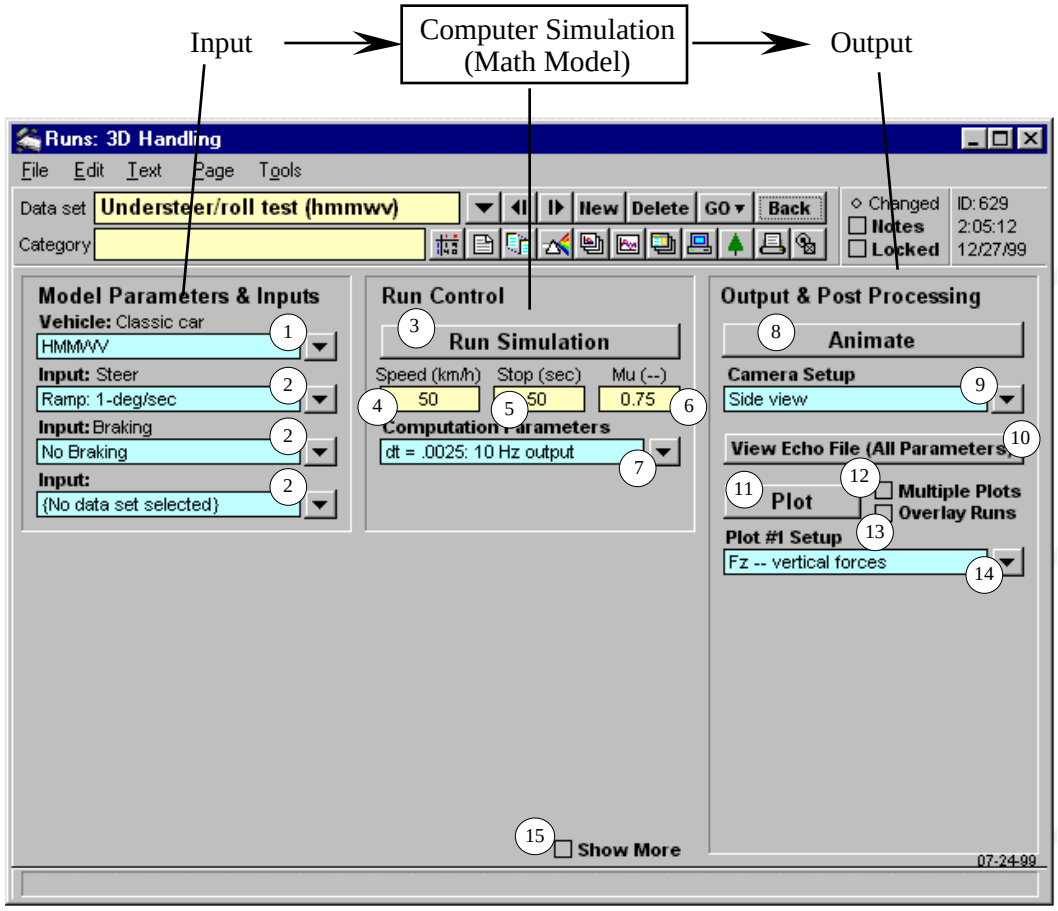


Figure 9.9. The Runs screen (simple view).

> Note: The **Runs** Screen has several possible appearances depending on whether various check boxes are checked. Figure 9.9 shows the simplest display. The most complex is shown later, after the basic controls have been described.

On the other hand, settings in the right-most region have no use until **after** a run is made. The linked data sets control the appearances of plots and animations. Changes in data in this region do not affect a simulation. They only affect how the simulation results are viewed.

Priorities of Data Links

The main purpose of this screen is to set up conditions for a run. It is possible to have conflicting information. For example, you might specify a speed in the yellow field (4) and also specify a throttle with one of the input links (2). Which takes priority?

The solver programs in CarSimEd handle inputs very simply: each line of input updates the simulation description. When there are conflicting inputs, the last input read is the one used. Therefore, the priorities for the inputs specified in the **Runs** screen are determined completely by the order in which they are sent to the solver programs. The first ones have the lowest priority, the last ones have the highest priority.

Here is the order in which the data from fields and links in the screen are sent to the solver programs. (See Figure 9.10 for the detailed view with all the links.)

1. The **Based On** link (18) is first, and therefore has the lowest priority.

> Note: This link is visible only when the Show More box (15) is checked (see Figure 9.10).

1. The **Vehicle** link (1).
2. The **Input** links on the left side of the screen (2), in a top-down sequence. Thus, those at the top are first and have lower priority than those at the bottom.
3. The **Computation Parameters** link (7).
4. The Speed (4) and Stop fields (5).
5. The **Misc. Data** field (17).

> Note: This link is visible only when the Show More box (15) is checked (see Figure 9.10).

1. The **Overriding Data (from Batch)** link (19) is last, and therefore has the highest priority.

Notes: To see the details of how the inputs from the screen are passed to the solver program, view ParsTree (use the **Tools** menu item **View ParsTree** or click the button .

This link is visible only when the Show More box (15) is checked.

User Settings and Controls (Simple Display)

(1) Link to vehicle data set from the library **Vehicles: Car**. The type of vehicle is shown as a sub-title above the blue field. In the figure, the type is `car`.

Note: In CarSimEd there is just one kind of 3D car model. However, in the commercial version there are several kinds of vehicles and the type is shown here.

(2) Links to screens for inputs and disturbances. The main inputs used in CarSimEd are braking, steering, and throttle.

(3) **Run Simulation** button. This is the main button in CarSimEd. Click to run the appropriate vehicle solver program using the current model parameters and inputs.

(4) Speed (keyword = `SPEED`). The vehicle model applies a closed-loop speed controller to maintain this speed until the brakes are applied. Alternatively, one of the input fields (2) can be linked to a data set defining a throttle input, in which cas the speed is used only to set the initial condition for the simulated test.

(5) Simulation stop time (keyword = `STOPT`). The simulation normally runs until this time is reached. However, the 3D handling model will stop earlier under some other conditions:

- The vehicle roll angle exceeds a specified limit that implies that a rollover accident was inevitable (keyword = `ROLL_STOP`).
- The absolute vehicle speed dropped below a specified threshold (keyword = `V_STOP`).

The low-speed threshold (`V_STOP`) is set on the **Input: Braking** screen. The other parameter (or both) can be optionally set using the **Misc. Data** field (17) described below.

(6) Road-tire friction parameter (keyword = `MU`). This parameter is used to scale the tire forces when the road-tire friction parameter, generally called μ, to be used for a run is different than the μ of the testing equipment used to generate the tables of the selected tire's cornering stiffness and pneumatic trail.

(7) **Computation Parameters** link. The linked data set specifies the integration time step, the print interval, the type of integration, and related parameters. For details, see the section **Computation Parameters** in this chapter.

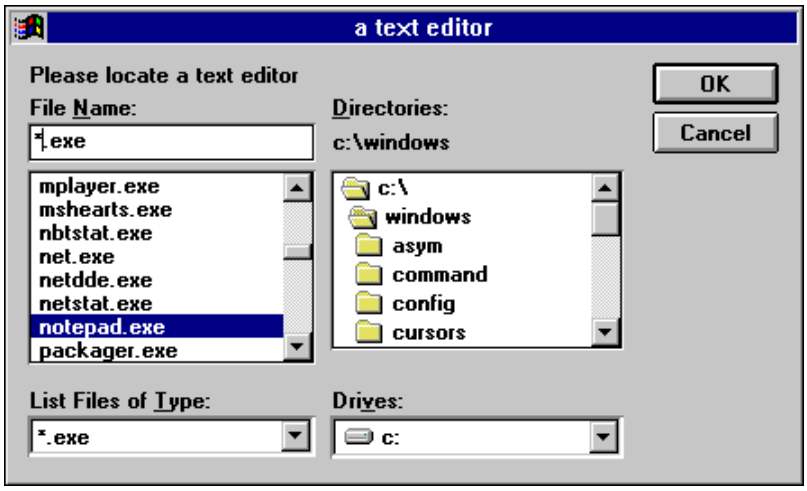
(8) **Animate** button. Click to run the wire-frame animator program and view motions of the vehicle as predicted by the simulation. An error message will be printed if the output file does not exist, which usually indicates that the run has not yet been made.

(9) **Camera Setup** link. The linked data establish the camera parameters and motion. For details see the section in this chapter called **Animator: Camera Setup**.

(10) **View Echo File (All Parameters)** button. This button opens a text editor with an echo file produced by the solver program.

The echo file has the extension LPF and is similar to the file listed in Appendix F. If the run has not yet been made, or if it aborted without writing the LPF file, then the text editor will show a blank window.

If you click this button and get an error message or a file browser dialog as shown below, then CarSimEd could not find the text editor.



The default text editor is a program called WinVI (Winvi32.exe, located in the `Programs` directory). If you want to change the default text editor, you can Control-click this button to bring up the file dialog and choose a different text editor.

(11) **Plot** button. Click to view a plot of output variables calculated during the simulation run. The plots are drawn by the WinEP program. An error message will be printed if the output file does not exist (for example, if the run has not yet been made).

Up to four separate plots can be automatically generated (each with many variables, taken from up to three runs). In order to generate more than one plot, the **Multiple Plots** box (12) must be checked.

The plot(s) shown by the plotter are defined by the linked plot data sets (14).

(12) **Multiple Plots** check box. This check box has two effects:

1. When checked it displays three plot setup links in addition to the first one (14).
2. When checked, it allows you to create up to four plots with a single click of the **Plot** button (11).

If the box is not checked, the additional plot links are not cleared. They are just hidden and are not used.

(13) **Overlay Runs** check box. This check box has two effects:

1. When checked it displays two links for other runs or ERD files.
2. When checked, it allows data from up to three files to be overlaid. (One file is the output associated with the current **Runs** data set. The other two are specified with the additional two links.)

If the box is not checked, the additional links are not cleared, they are just hidden.

(14) **Plot Setup** link. The linked data controls what information will be extracted from the output file and how that information will be displayed. Up to four plot setup links can be set, to create four independent plots at a time.

Only the first link is displayed if the **Multiple Plots** box (12) is not checked.

(15) **Show More** check box. This check box has a single effect: it shows more control objects, as shown in Figure 9.10.

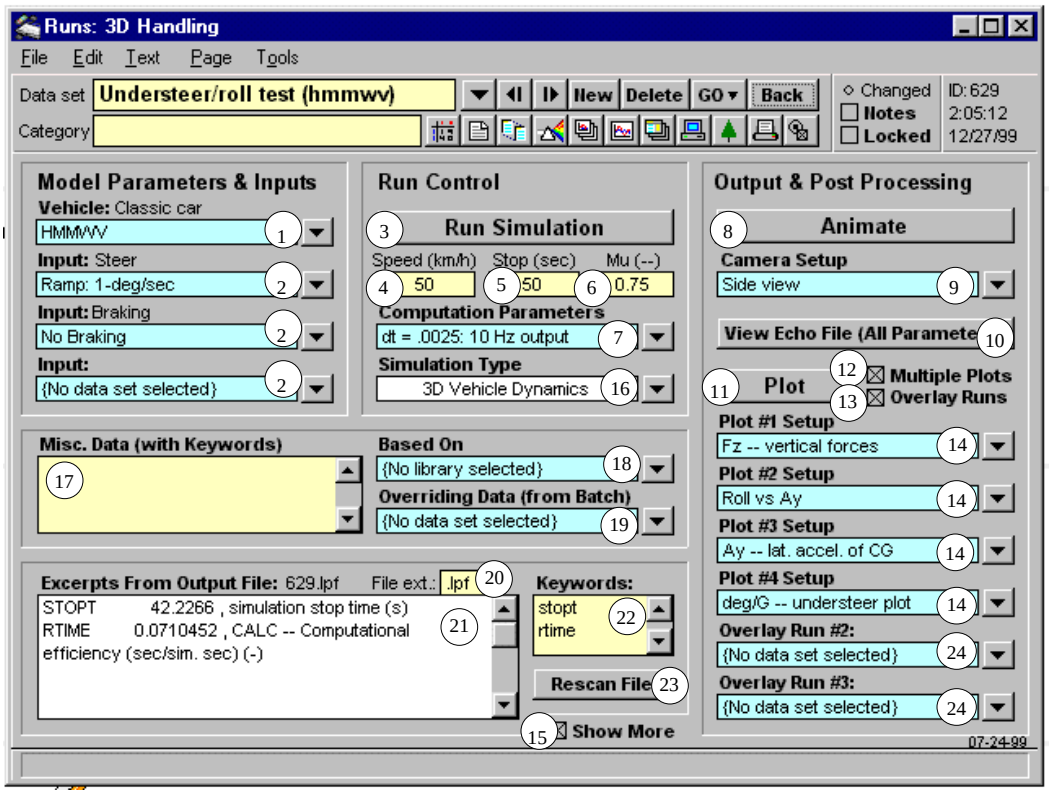


Figure 9.10. Runs screen with all controls showing.

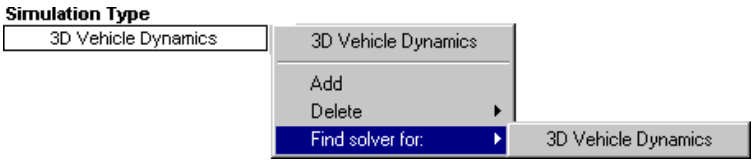
Note: Figure 9.10 shows the **Runs** screen with all three display boxes checked ((12),(13), and (15)). All possible data fields and links are displayed.

Additional User Settings and Controls

(16) **Simulation Type** link. This shows the current type of simulation model. As installed, CarSimEd has a single type. (Custom versions may have modified or extended models.)

The triangle button displays a pull-down menu with options for adding and deleting simulation types, and for locating a solver program (extension EXE). Unless you add new solver programs, you should never use this menu. In fact, the menu is disabled unless the check box for **Advanced Users** is checked in the **Preferences** screen. (To access the preferences, use the **Tools** menu item **Preferences** or click the button in the ribbon bar.)

When you click the **Run Simulation** button (3), the CarSimEd database launches the EXE file associated with the type of vehicle indicated above (1) and the simulation type.



The menu lists all of the installed simulation types, followed by three utility commands.

> Note: CarSimEd supports multiple solver programs, and this link is the interface to them. When you click the **Run Simulation** button, CarSimEd searches a hidden table for an entry containing the current type of vehicle displayed above the blue link (1) (in Figure 9.10 the vehicle type is `independent`) and also the **Simulation Type** (in Figure 9.10 the simulation type is `3D Vehicle Dynamics`). The table contains a corresponding pathname (e.g., `Programs\3d_car.exe`) for the solver program associated with the vehicle type and simulation type.

When new versions of solver programs are made, they can be added to the system without replacing the old ones, by using this pull-down menu to add a new simulation type. The menu also has options for deleting a simulation type, and for specifying exactly which EXE file will be run when you click the **Run Simulation** button.

CarSimEd allows you to associate EXE files and PIFs (Program Information Files) with the **Run Simulation** button. If you want to link to a software package that requires arguments, the way to do it is to make a PIF with the desired properties and then link to that. When the Windows file dialog box comes up, it is set to show only EXE files. If you want to link to a PIF, change the extension in the dialog box from `*.EXE` to `*.PIF` in order to see the PIFs.

See the Chapter 5 section **Changing A Solver Program** for instructions on using this menu.

(17) **Misc. Data** field. This is a field where any parameter can be set. The format for each line of text should consist of a parameter name, then a blank space, and then the parameter value. This field can contain any text that would be recognized by the solver program. See Appendix F for a list of all the keywords and parameters that can be specified.

This link is provided for special applications. (Most users never need it.) A few cases where you might consider using it are:

- to override a parameter to perform a quick "what if" run without making new data sets in other libraries;
- to set some of the seldom-used parameters that are not contained in any of the CarSimEd libraries (for example, the maximum roll angle that is allowed before the run is ended can be set here);
- to continue a run, perhaps after changing one or more model parameters (see Chapter 5 section **Continuing a Run**);
- to make a run using a vehicle description whose parameter values have been lost (the **based on** method reads from an LPF echo file, rather than the normal PAR files associated with the SGUI screens); or
- for certain debugging operations, such as confirming that the parameters specified in the screen are being processed by the solver program.

(18) **Based On** link. This link is sometimes used to base a new run on the data from an old run. If a new run is made, all model parameters and inputs from the old run will be read by the solver program before any of the inputs from this data screen are read. The final conditions from the other run are also read, which means that the new run will continue where the old one left off. Any inputs specified on this page will overwrite the data from the previous run.

The Chapter 5 section **Continuing a Run** explains some of the uses of this link.

(19) **Overriding Data** link. This link is used by the system when batch runs are made using the **Runs: Batch** library. If a run is made under batch control, this link can be used to go to the batch data set. (This is described in more detail in the section for the **Runs: Batch** screen.) As shown by the sequence of items on page 190, data from this link override everything else on the screen except the **Misc. Data** field (17).

This blue field is normally not linked to anything unless a batch run has been made.

(20) File extension. This determines which echo file is scanned when a run is made or the **Rescan File** button is clicked. The normal values are LPF (scan the file made with final conditions), or LPO (scan the file made with initial conditions).

(21) **Excerpts from Output File**. This field shows selected lines scanned from one of the output files. It is updated whenever a run is made or when the **Rescan** button (23) is clicked. This field is provided (along with the supporting items (20), (22), and (23)) to rapidly access information from the echo files produced when you make a run. For example, you can show the final value of a state variable, such as distance traveled, roll angle, etc.

To do this, look through an echo file (click the button **View Echo File (All Parameters)** (10)). Any keyword from the file can be placed into the keywords list (22), and lines beginning with those keywords will be placed into this field.

(22) **Keywords** field. Each line in this field is a keyword that will be used to scan an output echo file whenever a run is made or when the **Rescan** button is clicked.

(23) **Rescan** button. Click this button to cause CarSimEd to scan a file for lines beginning with the keywords listed in the **Keywords** field (22). Those lines are copied into the adjacent white field (21). The file that is scanned has a root name matching the ID of the current data set and the specified extension (20). This is done automatically when a new run is made. However, you can click this button to force the program to scan the file immediately. This is done mainly after changing the contents of the keywords field (22).

(24) **Overlay Run** links. These links are used to generate overlay plots, in which variables from the current simulation run are plotted with the same variables from other files. To overlay variables from two or three simulation runs, link to this library (**Runs**) and then pick another run from which to extract variables to plot. To overlay variables with ERD files not from this library (for example, measured test data), use the menu command to open an ERD file.

Location in CarSimEd

The **Runs** screen can be accessed from the **Tools** menu and the ribbon bar button: . It can also be accessed from the CarSimEd Startup screen:

CarSimEd Startup
  Runs: 3D Handling

File Location

`Runs_3D\Runs_3D.tbk`

Runs: SIMULINK CMEX Version

The **Runs** screen is central to the user interface in CarSimEd. It is used to set up simulation runs and to view results with post-processing programs.

> Note: The **Runs** Screen has several possible appearances depending on whether various check boxes are checked. Figure 9.11 shows the simplest display. The most complex is shown in the next section

Discussion

CarSimEd includes four **Runs** screens. Three are for running the stand-alone solver programs (3D Car, 2D Ride, and Suspensions Analysis) and the other is for running with SIMULINK. These screens are nearly identical in appearance and function. This section

describes two buttons that are unique to the SIMULINK version, along with the links to the vehicle and simulation type. (These are the four numbered items in Figure 9.11.)

All other features are the same as in the stand-alone version and are described in the next section, **Runs: 3D Handling**.

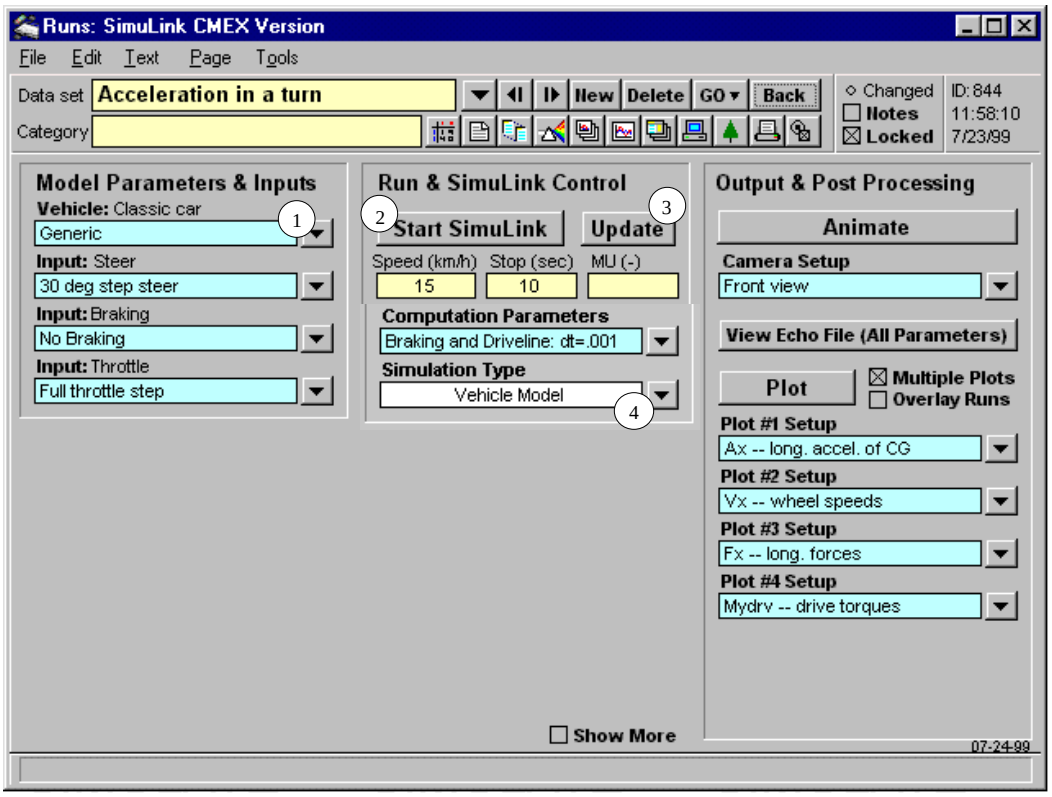


Figure 9.11. The Runs screen for SIMULINK.

User Settings

(1) Link to vehicle data set from the library **Vehicles: Car**. The type of vehicle is shown as a sub-title above the blue field. In the figure, the type is `independent`.

(2) **Start SimuLink** button. This is the main button for the screen. Click to launch SIMULINK with the model associated with the selected **Vehicle** (1) and **Simulation Type** (4). *Use this if SIMULINK and MATLAB are not already running*.

> Note: As described in Chapter 2, this button cannot be used unless MATLAB and SIMULINK are installed on your machine. (They are not part of CarSimEd, and must be licensed separately from The MathWorks, Inc.)

After clicking this button, you should see the model in SIMULINK. Type Ctrl+T to start the run. Or, select the **Start** menu item from the **Simulate** menu. When the simulation

runs, the CarSimEd module (a DLL file containing a CMEX S-function with the car model) reads all inputs from the CarSimEd libraries and proceeds. When the run is completed, you can view plots and animations from the **Runs** screen. Also, you can use any of the MATLAB analysis tools.

After completing a run, you can quit MATLAB to return to this screen, or you can leave MATLAB and SIMULINK open, and return to this screen by typing Alt-Tab or by using the Windows Task bar.

If you will be making many runs using the same SIMULINK model, it is not necessary to exit MATLAB. The new runs can be set up using the **Update** button (3).

However, if you wish to make a new run using a different SIMULINK model, it is best to exit MATLAB before initiating the next run.

(3) **Update** button. This button does half the work of the **Start Simulink** button (2). It causes CarSimEd to update the information that will be read by SIMULINK, but it does not launch SIMULINK. If SIMULINK is already open with the model you are using, you can use this button to update the information that will be read by the program when it runs.

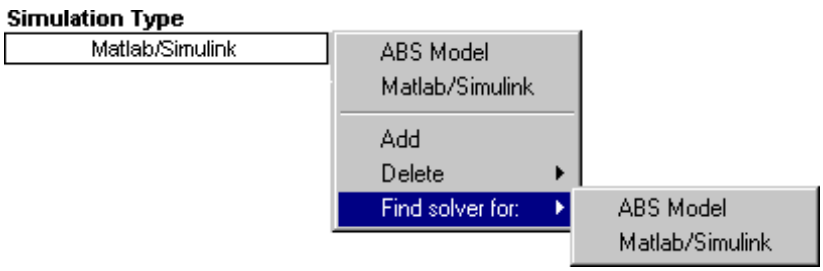
In a typical CarSimEd/SIMULINK session, you will use the **Start Simulink** button (2) to make the first run, and then use the **Update** button for all subsequent runs.

(4) **Simulaton Type** link. This shows the current type of SIMULINK model. As installed, CarSimEd has two example SIMULINK models. Once you start using CarSimEd with SIMULINK, it is likely you will add more SIMULINK models.

The triangle button displays a pull-down menu with options for adding and deleting simulation types, and for locating a folder with the necessary SIMULINK files. Unless you add new SIMULINK models, you should never use this menu. In fact, the menu is disabled unless the check box for **Advanced Users** is checked in the **Preferences** screen. (To access the preferences, use the **Tools** menu item **Preferences** or click the button in the ribbon bar.)

As described in Chapter 2, each SIMULINK model resides in its own folder. The folder has a program information file (PIF) that is associated with the MATLAB program.

When you click the **Start Simulink** button (2) or the **Update** button (3), the CarSimEd database writes the simulation control file (SIMFILE) into the folder associated with both the current simulation type and the vehicle type.



The menu lists all of the installed simulation types, followed by three utility commands.

> Note: CarSimEd supports multiple solver programs, and this link is the interface to them. When you click **Start Simulink** or **Update** buttons, CarSimEd searches a hidden table for an entry containing the current *vehicle type* displayed above the blue link (1) (in Figure 9.11 the vehicle type is `independent`) and also the *simulation type* (**Simulation Type** in Figure 9.11, where the simulation type is `Vehicle Model`). The table contains a corresponding pathname (e.g., `matlab\is_cmex\matlab.pif`) for the MATLAB PIF associated with the vehicle type and simulation type.

When new SIMULINK models are made, they can be added to the system without replacing the old ones by using the pull-down menu (4) to add a new simulation type. The menu also has options for deleting a simulation type, and for specifying exactly which file will be run when you click the **Start Simulink** button.

When you use the Add menu item to link to a new MATLAB PIF, change the extension in the dialog box from `*.EXE` to `*.PIF` in order to see the PIFs.

See the Chapter 4 section **Changing A SIMULINK Model** for instructions on using this menu.

Location in CarSimEd

The **Runs: SIMULINK CMEX Version** screen can always be accessed from the **GO** menu and the CarSimEd Startup screen:

CarSimEd Startup
  Runs_cmx

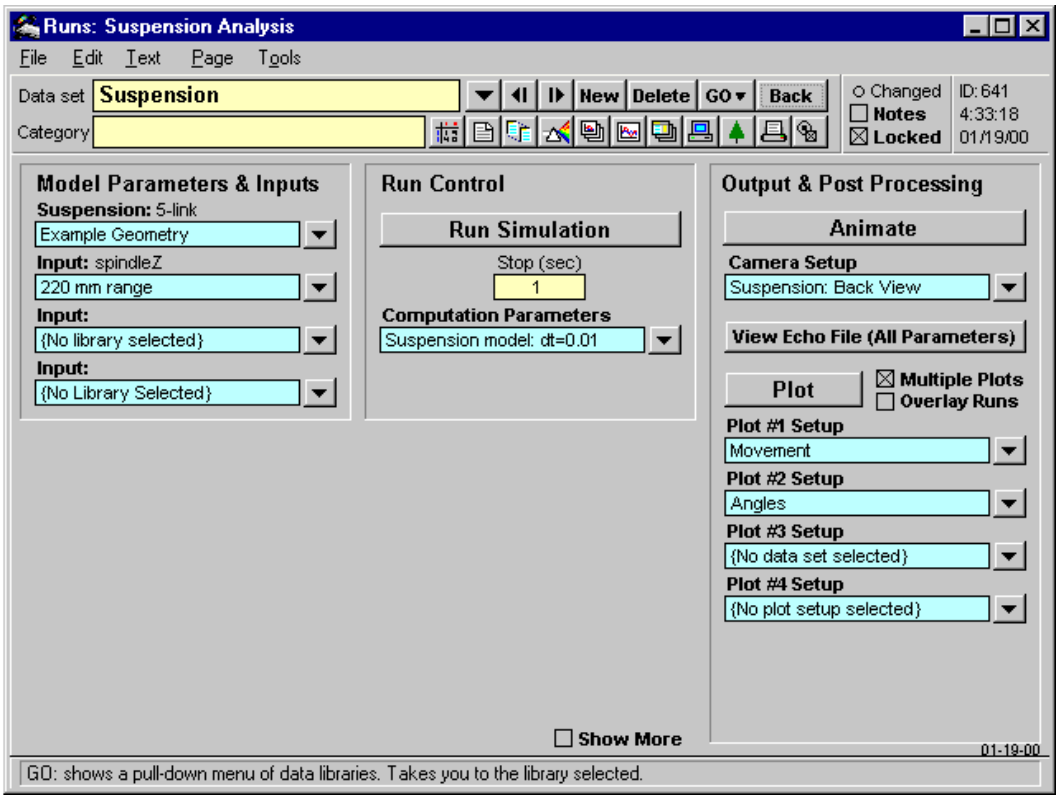
The **Tools** menu and the button in the ribbon bar will take you to one of the **Runs** libraries in CarSimEd. As installed, these are shortcuts for getting to the stand-alone **Runs** screen. If you primarily use the SIMULINK model (rather than the stand-alone solvers), you can change these shortcuts by Control-clicking the button in the ribbon bar. This brings up the file browser dialog box, which you can use to identify the file `Runs_cmx\Runs_cmx.tbk` as the default **Runs** library.

File Location

`Runs_cmx\Runs_cmx.tbk`

Runs: Suspension Analyses

The **Runs** screen is central to the user interface in CarSimEd. It is used to set up simulation runs and to view results with post-processing programs.



Discussion

CarSimEd includes four **Runs** screens. Three are for running the stand-alone solver programs (3D Car, 2D Ride, and Suspensions Analysis) and the other is for running with SIMULINK. These screens are nearly identical in appearance and function. The Suspension screen is nearly identical to the 3D Car screen (see Figure 9.9 on page 189). The differences are:

1. The suspension screen does not have yellow fields for speed and friction, because those parameters are not used in the suspension model.
2. The screen has a different title.
3. The system label is **Suspension** rather than **Vehicle**.

Please refer to the section for the 3D Car version for descriptions of the controls and settings of this **Runs** screen.

> Note: The only meaningful input for the suspension model is the spindle height as a function of time. Therefore, discussions of other inputs do not apply when using the suspension model.

Location in CarSimEd

The **Runs: Suspension Analyses** screen can always be accessed from the **GO** menu and the CarSimEd Startup screen:

CarSimEd Startup
Runs: Suspension Analyses

The **Tools** menu and the button in the ribbon bar will take you to one of the **Runs** libraries in CarSimEd. As installed, these are shortcuts for getting to the stand-alone **Runs** screen. If you primarily use the suspension model, you can change these shortcuts by Control-clicking the button in the ribbon bar. This brings up the file browser dialog box, which you can use to identify the file `Runs_sus\Runs_sus.tbk` as the default **Runs** library.

File Location

`Runs_sus\Runs_sus.tbk`

Runs: Batch

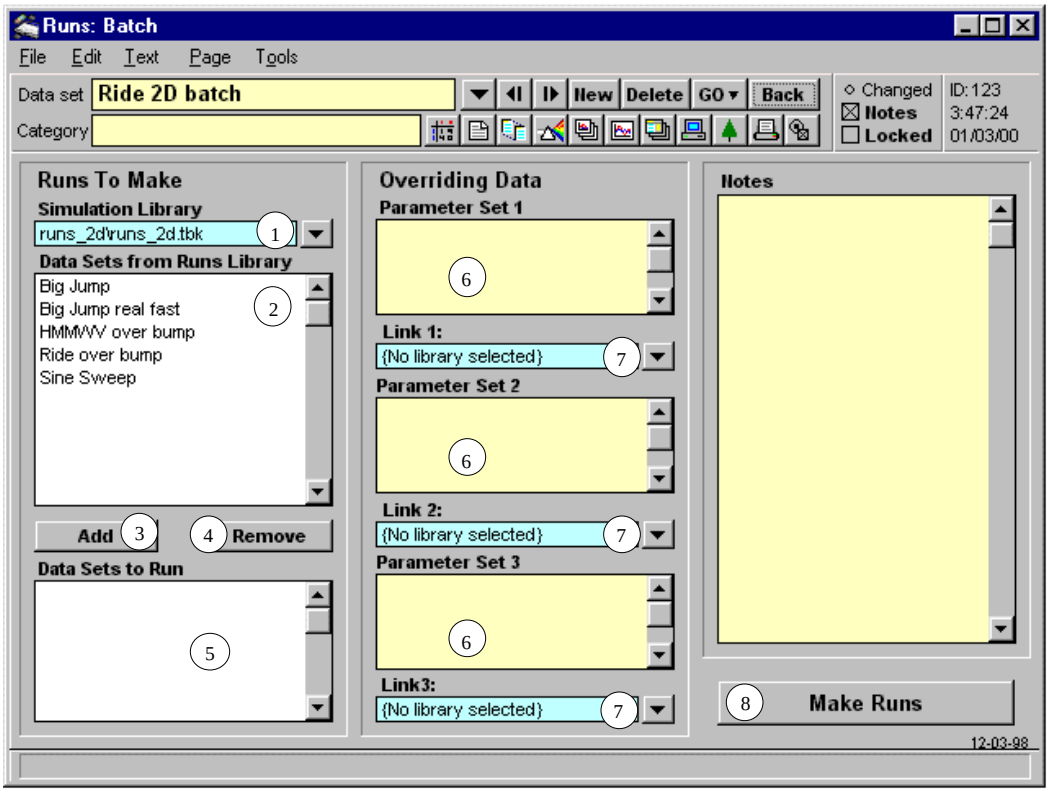
The batch runs screen is used to set up several runs ahead of time and make them all at once. It also can be used to override parameters.

Discussion

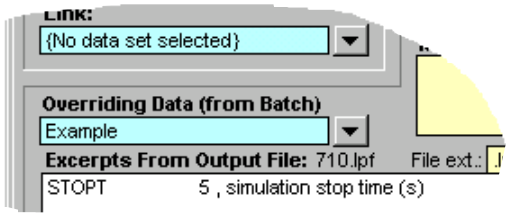
This screen is used to set up runs and then execute them in batch mode. It is handy for redoing a group of runs, say, because a change was made in a vehicle parameter value. It can also be used to change a parameter such as test speed, and make a set of runs again using the new parameter.

The basic method is:

1. Create a list of runs to be made (3) using existing data sets (2) from the **Runs** library.
2. Define parameters, and, as necessary, link to other data sets to override the conditions specified in the **Runs** data sets ((6), (7)).
3. Click the **Make Runs** button (8).



If no overriding data values or links are provided in step 2, the effect is exactly the same as going to each of the data sets from the **Runs** library and making the run again. However, if any parameters are specified or links are made, each data set in the **Runs** library is modified to use the overriding data. A link called **Overriding Data (from Batch)** is made to this data set. For example, in the figure above, the name of the data set is `Example`. If any runs were made, each affected run would have the following link set automatically:



This link is made so you can get from the runs data set back to the batch data set and see what parameters and links were used when the run was made.

User Settings and Controls

(1) **Simulation Library** link. For the batch run option to work, a link is needed to an existing **Runs** library (e.g., `Runs\Runs.tbk`). When this data set is opened, the linked library is automatically scanned by CarSimEd, and the names of all data sets are listed in the

field below (2). The adjacent triangle button has a pull-down menu that can be used to link to a different **Runs** library.

(2) The **Data Sets From Runs Library** list. This is a list of all of the data sets in the **Runs** library whose local pathname is shown above (1). Any line in brackets (<>) indicates a category heading.

Double-click on a line to add that run to the list named **Data Sets to Run** (5). This is the same as selecting the line and then clicking the **Add** button (3). Double-click on a line in brackets (<>) to add all runs in the category to the list (5).

(3) The **Add** button. Click to add the currently highlighted lines from the **Data Sets From Runs Library** list (2) to the **Data Sets to Run** list (5). Shift-click to add all runs.

(4) The **Remove** button. Click to remove highlighted data sets from the **Data Sets to Run** list (5). Shift-click to remove all.

> Note: The **Remove** button only affects the list of **Data Sets to Run** (5). This operation does not delete data sets from the database.

(5) The **Data Sets to Run** list. This is a list of the selected data sets from the **Runs** library. These are the data sets that will be run using the overriding data to the right ((6), (7)). Any line in brackets (<>) indicates a category heading.

Double-click on a line to remove it from the list. This is the same as selecting the line and then clicking the **Remove** button (4). Double clicking on a line in brackets will remove the entire category.

(6) **Parameter Sets (1, 2, and 3)**. These lists hold overriding data. If used, each line typically has a keyword followed by a value. For example, to set the speed to 60 km/h, enter the line `speed 60`. See Appendix F for the keywords recognized by CarSimEd models.

Any parameters specified in these fields will be used instead of the ones referenced in the **Runs** data set, or in data sets linked to the **Runs** data set.

This field should be left blank if you do not want to modify the input parameters or model descriptions for the runs to be made.

(7) **Links (1, 2, and 3)**. Use these links to apply data from any other CarSimEd library. If the data in the library can be applied to different parts of the vehicle (for example, a tire description can be applied to either the front or the rear axle of the vehicle), a reference to a part of the vehicle must be specified in the preceding **Parameter Set** (6) with an appropriate keyword (e.g., `iaxle 1`).

(8) The **Make Runs** button. Click to run all of the data sets listed in the **Data Sets to Run** list (5). Hold down the control key to interrupt the runs in progress.

Location in CarSimEd

Accessed from the **Tools** menu and the ribbon bar button:

File Location

`Batch\Runs_bat\Runs_bat.tbk`

Suspensions: Independent

Use this screen to specify the main properties of an independent suspension that affect the overall vehicle system response.

Discussion

In CarSimEd, an “independent” suspension is one in which vertical movement of one wheel does not cause noticeable movement of the other wheel if the anti-roll bar is disconnected. In contrast, a “solid axle” suspension has an actual axle or linkage system that cause both wheels to roll together. (CarSimEd only includes independent suspensions. The commercial CarSim package supports additional suspension types.)

Independent Suspension Properties

The kinematics of the suspension linkages are described in terms of how a wheel moves laterally and longitudinally as the suspension deflects vertically. The lateral movement affects the transfer of tire lateral force to the body and the resulting body roll. These effects are commonly described using the concept of a roll center. The mathematical models in CarSimEd do not include an actual roll center point—the wheels are assumed to move in a straight line as shown by the lines of motion in the kinematics section in Figure 9.12. However, the direction of the line of motion is specified with a roll center height, for compatibility with other models and data sets. (The CarSimEd model divides the roll center height by half the track to determine the amount the wheel moves laterally as the suspension compresses.)

The longitudinal movement has a similar effect involving pitch due to longitudinal tire force. As shown in Figure 9.12 for the lines of motion illustrating the ratio of longitudinal movement per unit of vertical movement of the wheel center.

Changes in steer due to compliance and suspension kinematics can drastically affect the response of a car with respect to steering inputs. The compliance is represented with linear coefficients. Kinematical effects are represented with nonlinear tables.

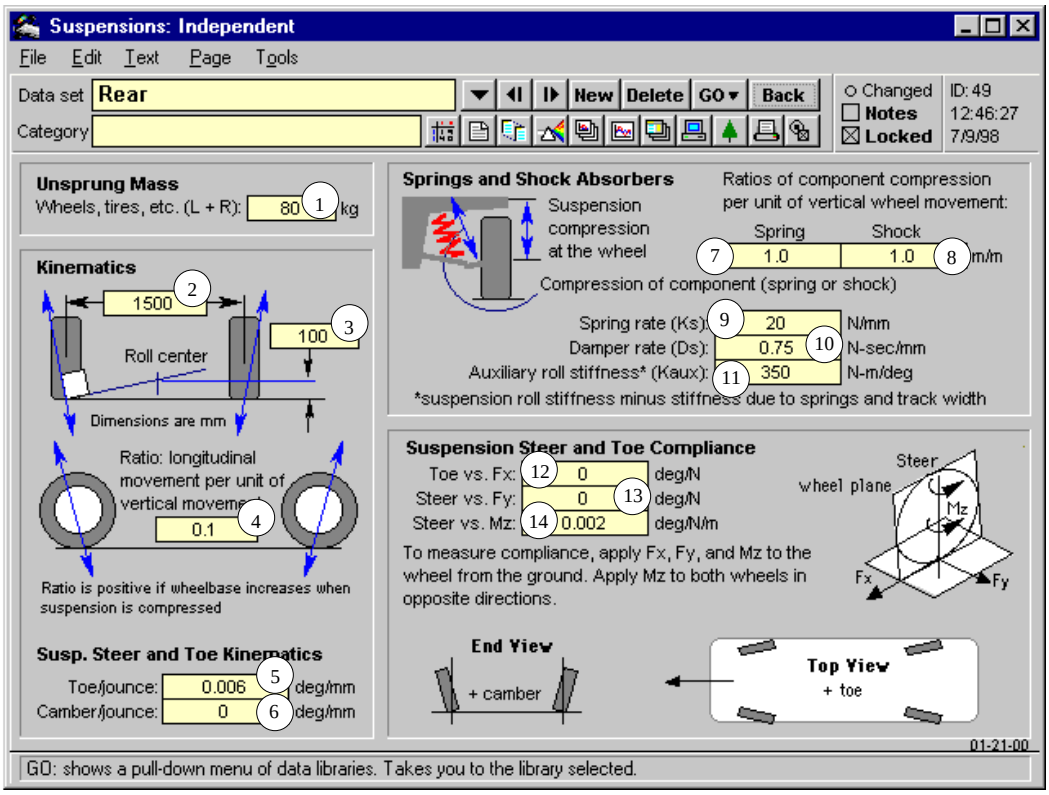

Suspensions: Independent
File Edit Text Page Tools
Data set Rear
Category
New Delete GO Back
Changed
Notes
Locked
ID: 49
12:46:27
7/9/98
Unsprung Mass
Wheels, tires, etc. (L + R): 80 1 kg
Kinematics
1500 2
100 3
Roll center
Dimensions are mm
Ratio: longitudinal movement per unit of vertical movement
0.1 4
Ratio is positive if wheelbase increases when suspension is compressed
Susp. Steer and Toe Kinematics
Toe/jounce: 0.006 5 deg/mm
Camber/jounce: 0 6 deg/mm
Springs and Shock Absorbers
Suspension compression at the wheel
Compression of component (spring or shock)
Ratios of component compression per unit of vertical wheel movement:
Spring Shock
7 1.0 1.0 8 m/m
Spring rate (Ks): 9 20 N/mm
Damper rate (Ds): 0.75 10 N-sec/mm
Auxiliary roll stiffness* (Kaux): 11 350 N-m/deg
*suspension roll stiffness minus stiffness due to springs and track width
Suspension Steer and Toe Compliance
Toe vs. Fx: 12 0 deg/N
Steer vs. Fy: 0 13 deg/N
Steer vs. Mz: 14 0.002 deg/N/m
To measure compliance, apply Fx, Fy, and Mz to the wheel from the ground. Apply Mz to both wheels in opposite directions.
Steer
wheel plane
Mz
Fx
Fy
End View
+ camber
Top View
+ toe
01-21-00
GO: shows a pull-down menu of data libraries. Takes you to the library selected.

Figure 9.12. Independent suspension screen.

User Settings

(1) Unsprung mass (keyword = `MUS (IAXLE)`). Mass of wheels, tires, brakes, and all parts that generally move vertically with the wheel as the suspension deflects. For parts such as driveline components and suspension linkages that have one end attached to the moving wheel and the other to the sprung mass, you can add about half their masses to the overall unsprung mass. This value includes both wheels in the suspension.

(2) Track width (keyword = `LTK (IAXLE)`). Lateral distance between centers of tire contact at the simulation load condition.

(3) Height of axle roll center above ground at the simulation load condition (keyword = `HRC (IAXLE)`). This is typically a value between 0 and 200 mm for independent suspensions.

The mathematical models in CarSim do not include an actual roll center point—the wheels are assumed to move in a straight line as shown by the lines of motion in the kinematics section in Figure 9.12. The direction of the line of motion is specified with a roll center height solely to maintain compatibility with other models and existing data.

(4) Wheelbase change per unit jounce (keyword = `RAP (IAXLE)`). This should be a positive number for both front and rear suspensions, with a value between 0.05 and 0.25 mm/mm.

> Note: The sign convention, shown in Figure 9.12, is designed such that a positive value means the wheel moves away from the center of the vehicle as it moves up. In the front, this is called anti-dive. In the rear, anti-squat.

(5) Ratio of toe change per unit of suspension compression (keyword = RTOE (IAXLE)). As shown in the figure, positive toe is steer to the inside (left steer for the right wheels, right steer for the left wheels). Suspension kinematics normally cause toe to vary in a complex way with suspension deflection. This coefficient should be selected to represent the linear behavior near the nominal suspension compression.

(6) Ratio of camber change per unit of suspension compression (keyword = RCAM (IAXLE)). As shown in the figure, positive camber is when the wheel leans out at the top. Suspension kinematics normally cause camber to decrease with suspension compression. Thus, this value is often negative.

(7) Ratio of spring jounce (compression) to jounce at wheel (keyword = `RSPRING (IAXLE)`). This determines the mechanical advantage of the suspension with respect to the spring. The value is typically between 0.5 (for some SLA suspensions) and 1.0 (for some MacPherson strut suspensions).

(8) Ratio of damper jounce to jounce at wheel (keyword = `RDAMP (IAXLE)`). This determines the mechanical advantage of the suspension with respect to the shock absorber. The value is typically between 0.5 (for some SLA suspensions) and 1.0 (for some MacPherson strut suspensions).

(9) Spring rate (keyword = `KS (IAXLE)`). This is the linear rate of a single suspension spring. You can take the rate at the spring, rather than at the wheel, because an additional parameter accounts for the mechanical advantage of the suspension (7).

(10) Damper rate (keyword = `DS (IAXLE)`). This is the linear rate of a single shock absorber. You can take the rate of the isolated damper, rather than at the wheel, because an additional parameter accounts for the mechanical advantage of the suspension (8).

(11) Auxiliary roll stiffness, including stabilizer bar (keyword = `KAUX (IAXLE)`). This coefficient is provided to account for the difference between the overall roll stiffness and the stiffness provided by the springs along. It is usually positive.

The overall roll stiffness of a suspension is the roll moment (N-m) needed to roll the vehicle body one degree, with the other suspension disconnected, and the tires replaced with rigid elements. The springs provide a certain amount of roll stiffness, based on the spring rate, the mechanical advantage of the suspension, and the track width. The total roll stiffness may differ from the rate calculated for the springs alone. Additional stiffness is provided by anti-sway bars and by over constrained suspension linkages.

On the other hand, compliance in the suspension can sometimes result in less roll stiffness than would be predicted from the spring rates. In this case you would specify a negative value for auxiliary stiffness.

(12) Coefficient for change in toe per change of tire longitudinal force (keyword = `CTFX (IAXLE)`). A forward tractive force tends to bend a suspension forward, steering the wheel inward (positive toe). Therefore this parameter is likely to have a small but positive value.

(13) Coefficient for change of steer angle per change of tire lateral force (keyword = `CSFY (IAXLE)`). For wheels which can be steered, the steer axis is usually inclined to intersect the ground in front of the center of tire contact. Thus, a positive lateral force (to the left), acting behind the steer axis, usually causes some steer to the right (negative). This coefficient is therefore likely to have a small negative value for a steered wheel. For rear wheels, is should have a value close to zero.

(14) Coefficient for change of steer angle per change of tire aligning torque (keyword = `CSMZ (IAXLE)`). The suspension elements usually deflect when a steering torque is applied to the wheel. Because the steer and moment have the same sign convention, the compliance coefficient is nearly always positive.

Steering as a result of aligning torque is due both to compliance in the suspension, and to compliance in the steering column. If equal and opposite steering torques are applied to both the left and right road wheels, the resulting deflection is mainly due to the suspension alone.

The suspension compliance is subtracted from the total steering compliance that would be measured for one wheel, to determine the compliance for the steering column (specified in the **Cars** data screen).

Location in CarSimEd

CarSimEd Startup
  Runs
    Vehicles: Car
      Suspension Systems: Car Suspension

File Location

`Vehicles\Susp_ed\Susp_ed.tbk`

Suspensions: 5-Link Independent

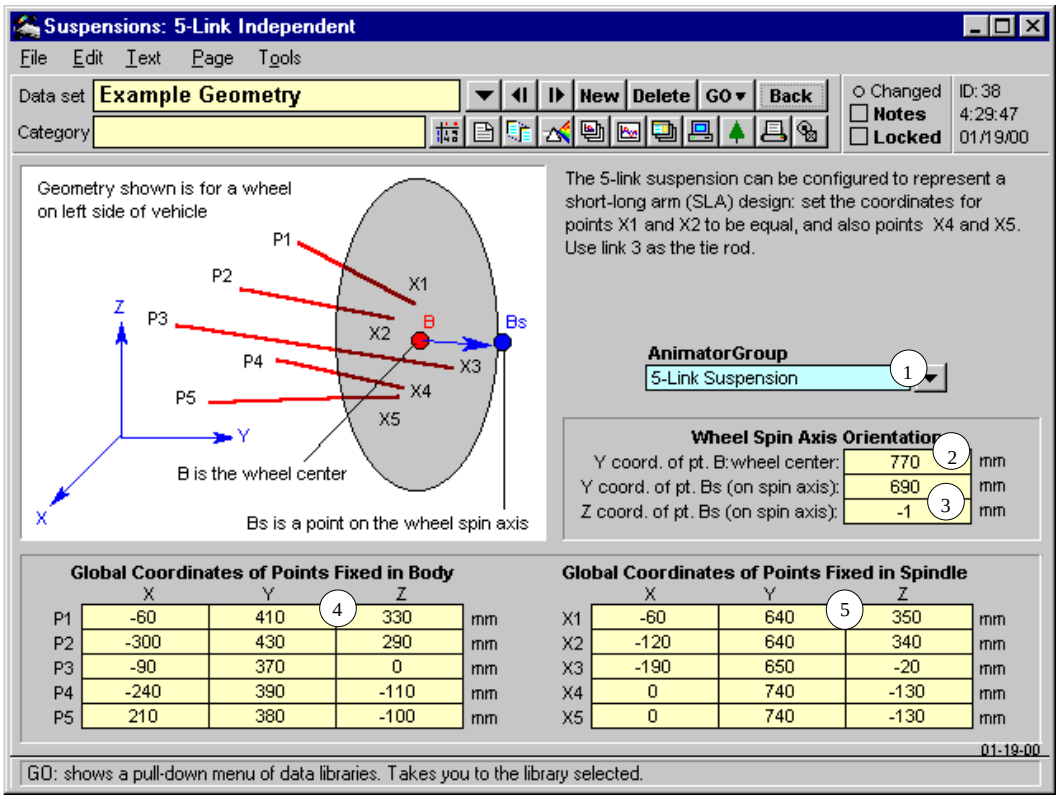
Use this screen to specify the geometry for a five-link suspension.

Discussion

CarSimEd includes a simulation program to compute changes in steer (toe) and camber as functions of vertical spindle position. In addition, the angles for all five links are computed, as are the X and Y coordinates of the wheel center.

Example results for this model have been published with the intention of providing a benchmark for validating new simulation codes.

M. Hiller and S. Frik, “Five-Link Suspension,” from *Multibody Computer Codes in Vehicle System Dynamics*, Supplement to *Vehicle System Dynamics*, vol. 22, 1993. pp 254 - 262.


Suspensions: 5-Link Independent
File Edit Text Page Tools
Data set Example Geometry
Category
New Delete GO Back
Changed
Notes
Locked
ID: 38
4:29:47
01/19/00
Geometry shown is for a wheel on left side of vehicle
P1
P2
P3
P4
P5
X1
X2
X3
X4
X5
B
Bs
Z
Y
X
B is the wheel center
Bs is a point on the wheel spin axis
The 5-link suspension can be configured to represent a short-long arm (SLA) design: set the coordinates for points X1 and X2 to be equal, and also points X4 and X5. Use link 3 as the tie rod.
AnimatorGroup
5-Link Suspension
1
Wheel Spin Axis Orientation
Y coord. of pt. B:wheel center: 770 mm
Y coord. of pt. Bs (on spin axis): 690 mm
Z coord. of pt. Bs (on spin axis): -1 mm
2
3
Global Coordinates of Points Fixed in Body
X Y Z
P1 -60 410 330 mm
P2 -300 430 290 mm
P3 -90 370 0 mm
P4 -240 390 -110 mm
P5 210 380 -100 mm
4
Global Coordinates of Points Fixed in Spindle
X Y Z
X1 -60 640 350 mm
X2 -120 640 340 mm
X3 -190 650 -20 mm
X4 0 740 -130 mm
X5 0 740 -130 mm
5
01-19-00
GO: shows a pull-down menu of data libraries. Takes you to the library selected.

User Settings

(1) Link to an animator group data set. The animator group should be set to provide a wire-frame matched to the suspension parts.

(2) Y coordinate of point B, the wheel center (keyword = BY). The X coordinate is defined as zero, and the Z coordinate is a variable that is varied during a run. The Y value changes during a run: the specified coordinate applies when the points (X1 to X5) are at their specified locations (5).

(3) Wheel spin orientation (keywords = BSY, BSZ). The orientation of the spin axis is defined by a second point, BS, lying outboard of point B. The Y and Z coordinates of this point determine how the spin axis is oriented when the points (X1 to X5) are at their specified locations (5).

(4) Coordinates of points fixed in the body (keywords = PX(1) - PX(5), PY(1) - PY(5), PZ(1) - PZ(5)).

(5) Coordinates of points fixed in the spindle body (keywords = XX(1) - XX(5), XY(1) - XY(5), XZ(1) - XZ(5)). These locations define a nominal configuration for the system.

Location in CarSimEd

CarSimEd Startup
  Runs: Suspension Analysis
    Link-Type Independent Suspension

File Location

`Vehicles\Susp_5L\Susp_5L.tbk`

Tires: CarSimEd Model

Use this screen to change the constants and links to the table look-ups which define the tire properties.

Discussion

This screen provides a location for you to enter a number of tire properties needed in the simulation. Tire vertical stiffness determines how the load varies as the tire bounces on the road. You can specify a vertical stiffness value here. If you are simulating the 2D vehicle ride response, this is the only tire parameter that will be used.

For small amounts of slip (lateral or longitudinal), the tire produces forces and aligning moment proportional to the slip. The coefficients are highly sensitive to load, and therefore the screen has links to other data screens that define the load sensitivity in tabular form.

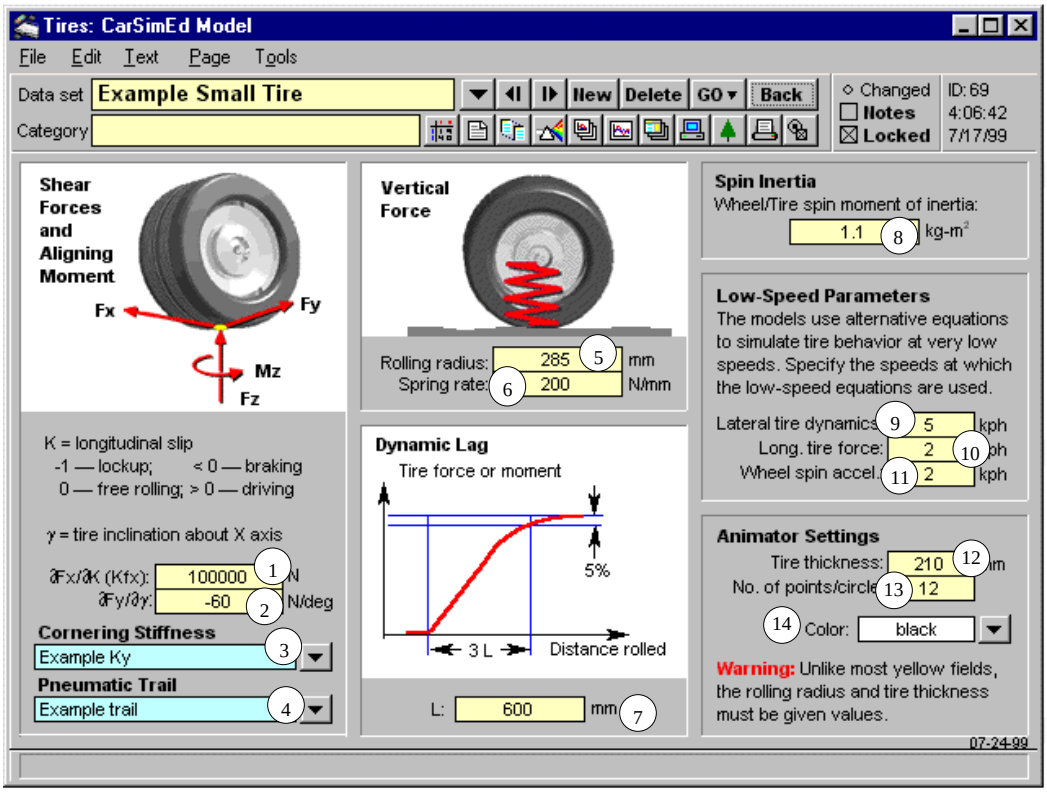
For braking, the dynamics of the spinning wheels are not of great interest. Without ABS simulation, the wheels generally reach a spin value in which the longitudinal tire force just balances the applied brake torque. The model uses a single coefficient to relate longitudinal slip to force.

For cornering, a quick equilibrium is not reached, and the relation between lateral slip and lateral force has a strong influence on the handling behavior of the vehicle. The CarSimEd tire model is described in Appendix G. It is based on an analysis of tire mechanics from Chapter 14 of the book *Race Car Vehicle Dynamics*, Milkier & Milkier, SAE, 1995.

When a tire experiences a slip angle, it does not immediately generate a lateral force, but must roll some distance to generate the lateral deflection necessary to sustain a force. Under a step steer the force builds up like a first-order lag in distance. You can input this distance, known as the relaxation length, on this screen.

At low speeds the determination of tire forces can become erratic because of numerical problems in calculating the slip conditions. In order to avoid these problems, you can specify a cut-off speed below which such mechanisms as relaxation length are modified.

All properties are specified for a single tire.



User Settings

(1) Longitudinal stiffness (keyword = KFX(IAXLE)). This coefficient defines the relationship between longitudinal slip and force for small amounts of slip. Although the coefficient is in reality sensitive to load, a constant value is used in the model because no matter what the stiffness is, the longitudinal force will balance the brake torque. The stiffness determines the amount of slip that occurs during that balance, and the longitudinal slip does affect the cornering behavior for combined slip (see Appendix G). However, given that the tire model is not adjustable in its combined slip behavior, a single value for K_{fx} is considered to be adequate for most conditions.

(2) Camber thrust coefficient (keyword = KFYCAM(IAXLE)). This coefficient defines the relationship between wheel inclination and side force. The sensitivity to inclination is an order of magnitude less than the sensitivity to slip (steer). Even though the coefficient is known to be sensitive to load, a constant value is used in the CarSimEd model due to its relatively small overall contribution to the tire force.

With the ISO coordinate system used in CarSimEd (shown in the screen figure), positive inclination (leaning to the right) causes negative lateral force (the force also goes to the right). Therefore, this coefficient should be negative.

> Note: With the SAE coordinate system this coefficient would be positive. Therefore, check the sign if you are obtaining data from a source where the SAE convention was used.

③ Link to cornering stiffness data. The linked data set specifies cornering stiffness as a function of load.

④ Link to cornering pneumatic trail data. The linked data set specifies pneumatic trail as a function of load.

⑤ Undeformed rolling radius (keyword = `HWC (IAXLE, ISIDE)`). This is the distance the tire rolls at zero load, divided by 2π. It is approximately the height of the wheel center when the tire is unloaded (lightly resting against the ground, fully inflated). It is used in the simulation model to relate rim speed of the tire to the forward speed at the wheel center. This number is larger than the loaded radius by an amount: F_Z / K_t (vertical force divided by linear spring rate).

⑥ Linear tire vertical spring rate (keyword = `KT (IAXLE, ISIDE)`). This coefficient is used in the model to calculate change in vertical load due to tire compression. The behavior is linear until the load reaches zero. Once the tire lifts off the ground, the force remains at zero unless contact is made again.

⑦ Tire relaxation length (keyword = `LRELAX (IAXLE)`). This is about one-third the distance that the tire must roll before side force due to slip angle builds up to its full value. It can be thought of as a spatial version of a time constant.

⑧ Spin moment of inertia of one wheel (keyword = `IW (IAXLE)`). This should include all parts that rotate with the wheel as a single rigid body, including the tire, brake rotors, and possibly driveline elements.

⑨ Cutoff speed for tire relaxation equations (keyword = `VLOW_ALPHA (IAXLE)`). The concept of relaxation length is valid for a range of speeds, but leads to strange behavior as speeds approach zero because the time constant associated with the dynamic lag goes to infinity. The time constant associated with relaxation is frozen when the speed drops below this value.

⑩ Cutoff speed for longitudinal tire forces (keyword = `VLOW_KAPPA (IAXLE)`). The classic model for predicting longitudinal force eliminates a direct dependency on speed by using a normalized longitudinal slip. However, the definition of longitudinal slip is singular at zero speed. At speeds approaching zero, the classic model can predict maximum tractive force, oscillating in the forward and rearward directions. This behavior is reduced in the CarSimEd model by attenuating the slip when the wheel is locked by the brakes and the speed drops below the level specified in this field (see Appendix G for details).

⑪ Cutoff speed for wheel spin acceleration (keyword = `VLOW_SPINA (IAXLE)`). Brake torque is the result of friction and always opposes the wheel spin. When the wheel spin approaches zero (lock up), a numerical instability can occur due to the reversal of the direction of the brake torque. To avoid numerical problems, the wheel spin equation is modified when the effective speed (spin multiplied by rolling radius) drops below the level specified in this field.

⑫ Thickness of wheel object drawn by wire-frame animator (keyword = `set_thickness`). This dimension is used for creating the wire-frame animation, but is

not used by the solver programs. Unlike most yellow fields, this must be given a value because the library automatically converts the units from mm to meters.

(13) Number of points in polygons used to represent the wheel in wire-frame animations (keyword = `set_num_points`). This value is not used by the solver programs.

(14) Color of wheel object drawn by wire-frame animator (keyword = `set_color`). Select the button next to the field to access a pull-down menu with the valid color names. This information is not used by the solver programs.

Location in CarSimEd

CarSimEd Startup
  Runs
    Vehicles: Car
      Tires: CarSimEd Model

File Location

`Vehicles\Tires_ed\Tires_ed.tbk`

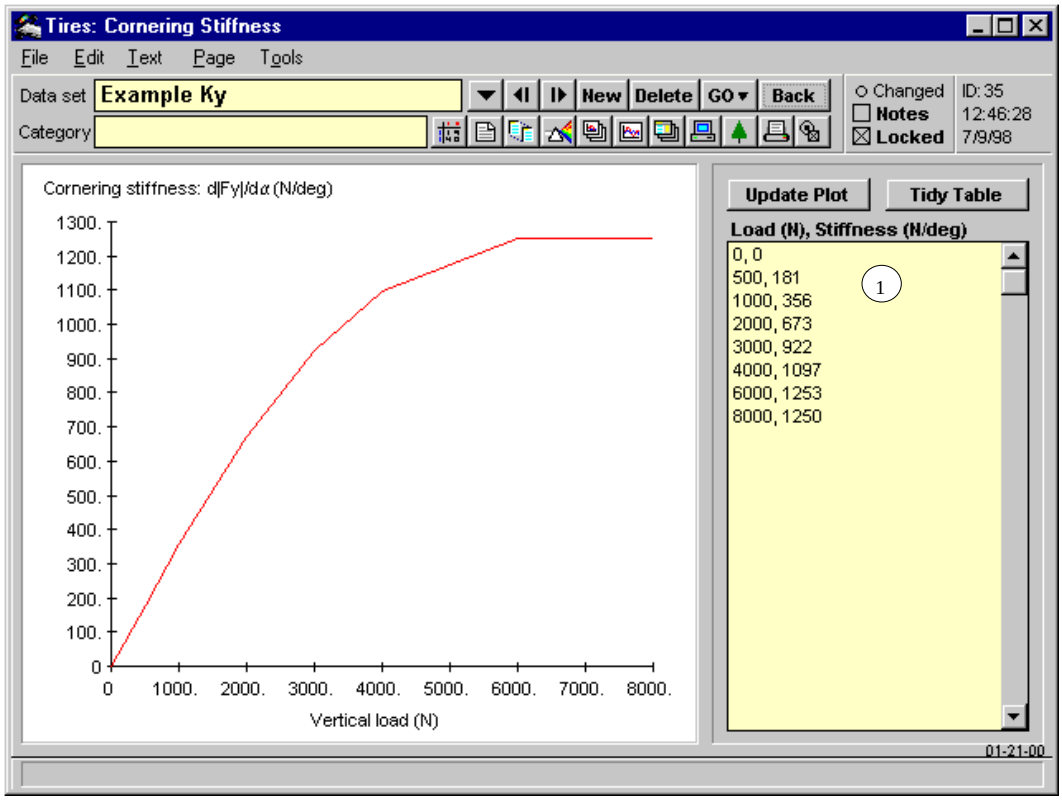
Tires: Cornering Stiffness

Use this screen to define the cornering stiffness of a tire as a function of vertical load.

Discussion

For small levels of lateral slip, tires produce lateral force that is proportional to negative slip angle. The coefficient is called cornering stiffness. It is strongly dependent on load. The CarSimEd model accounts for the load sensitivity with a tabular function.

For large slip angles, the CarSimEd model reduces the lateral force to account for friction limits. Details are provided in Appendix G.



User Settings

> Note: User settings that are common for all tabular data screens are described in Chapter 7, in the section **Tabular Data**.

(1) Table field for cornering stiffness (keyword = KFY_TABLE). Each line should have a value of cornering stiffness (N) followed by a corresponding value of trail (N/deg). Although lateral force and side slip have opposite signs, cornering stiffness is typically defined as a positive quantity, and therefore, all numbers in the table should be positive.

The solver program uses linear interpolation and flat-line extrapolation with this table. For values of load that are less than the range covered in the table, the first value of cornering stiffness is used. For values of load larger than the range covered, the last value of cornering stiffness is used. The table needs at least two lines of data or else an error message is generated.

Location in CarSimEd

CarSimEd Startup
  Runs
    Vehicles: Car
      Tires: CarSimEd
        Tires: Cornering Stiffness: Kfy

File Location

`Vehicles\Tires\Kfy\Kfy.tbk`

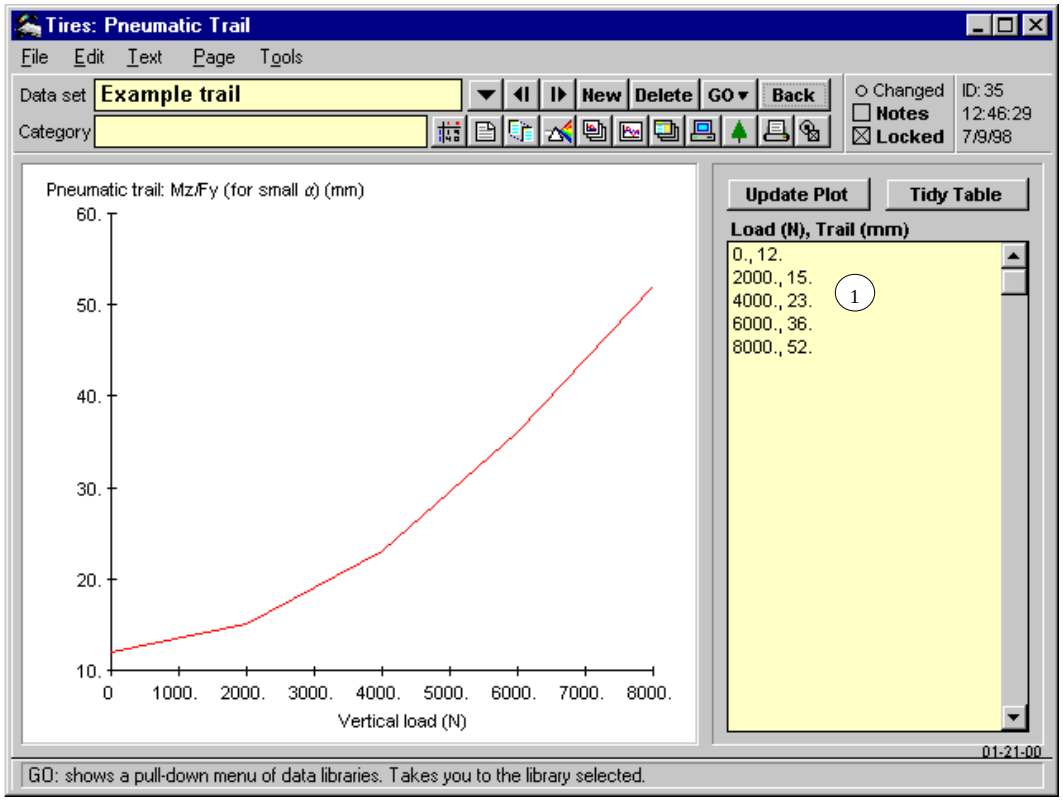
Tires: Pneumatic Trail

Use this screen to specify pneumatic trail as a function of vertical load.

Discussion

The line of action for the shear force produced by a tire is generally not exactly through the center of tire contact. For small lateral slip angles, the shear force usually acts behind the center. This distance acts as a moment arm, producing a torque that opposes the steering. It is called “self-aligning torque” or aligning moment.

The pneumatic trail is a function of the shape of the contact patch between the tire and road. Therefore, pneumatic trail is sensitive to inflation pressure and vertical load. This screen is used to describe this sensitivity to load.



User Settings

> Note: User settings that are common for all tabular data screens are described in Chapter 7, in the section **Tabular Data**.

(1) Table field for pneumatic trail (keyword = `TRAIL_TABLE`). Each line should have a value of load (N) followed by a corresponding value of trail (mm).

The solver program uses linear interpolation and flat-line extrapolation with this table. For values of load that are less than the range covered in the table, the first value of trail is used. For values of load larger than the range covered, the last value of trail is used. The table needs at least two lines of data or else an error message is generated.

Location in CarSimEd

CarSimEd Startup
  Runs
    Vehicles: Car
      Tires: CarSimEd
        Tires: Pneumatic Trail

File Location

`Vehicles\Tires\Trail\Trail.tbk`

Vehicles: Car

This screen is used to define the dimensions, mass properties, and component properties (links) for the vehicle used in both the 2D Ride and 3D Braking and Steering simulations.

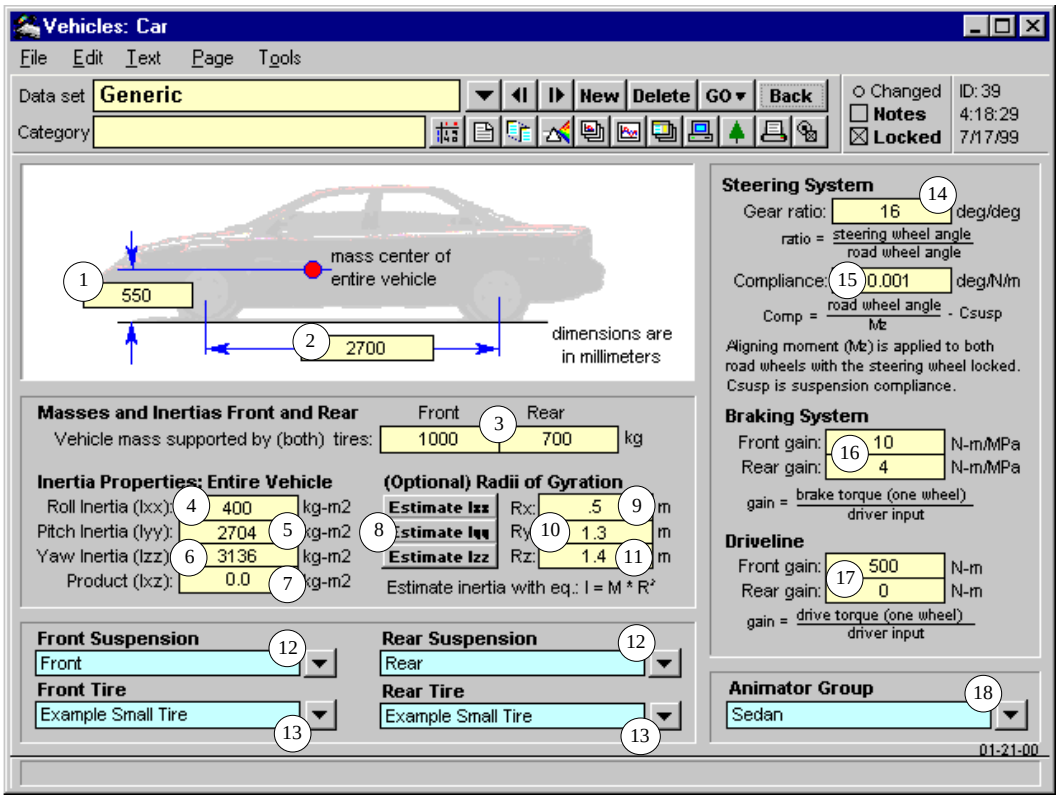
Discussion

One of the useful features of CarSimEd is that you can develop a whole fleet of vehicles but have to define the characteristics of each particular component only once. You can then use this particular component in any of your vehicles simply by linking to it with this screen. Furthermore, these same vehicles can be used in both the 2D Ride and 3D Braking and Steering simulations. (The 2D Ride solver program ignores the parameters associated with roll and yaw motion.)

User Settings

(1) Height of the mass center of the entire vehicle above the ground (keyword = `HCG`).

(2) Wheelbase (keyword = `LWB`). Longitudinal distance from the center of the front axle to the center of the rear axle. This value is also passed to the animator (keyword = `x_length`) to re-size the wire-frame shape longitudinally if the wheelbase is changed.



(3) Vehicle mass supported by wheels (keyword = `MF` for front two wheels and `MR` for rear two wheels). These are the masses that would be measured when the front and rear of the car are alternatively driven over a scale. These two values added together give the total mass of the vehicle.

(4) Roll Moment of Inertia of entire vehicle (keyword = `IXX`). The moment is taken about the mass center of the entire vehicle, normal to an X axis (longitudinal) that is parallel with the ground when the vehicle is at rest on a flat level surface. This value can be entered directly, or it can be calculated using an estimated radius of gyration (11) and the associated button **Estimate Ixx** (10).

(5) Pitch Moment of Inertia of entire vehicle (keyword = `IYY`). The moment is taken about the mass center of the entire vehicle, normal to a Y axis (lateral) that is parallel with the ground when the vehicle is at rest on a flat level surface. This value can be entered directly, or it can be calculated using an estimated radius of gyration (12) and the associated button **Estimate Iyy** (10).

(6) Yaw Moment of Inertia of entire vehicle (keyword = `IZZ`). The moment is taken about the mass center of the entire vehicle, normal to a Z axis (vertical) that is parallel with the gravity vector when the vehicle is at rest on a flat level surface. This value can be entered directly, or it can be calculated using an estimated radius of gyration (13) and the associated button **Estimate Izz** (10).

(7) X-Z Product of Inertia of entire vehicle (keyword = `IXZ`). The product is taken about the mass center of the entire vehicle, and is defined as the negative of the volume integral:

$$I_{XZ} = -\int_V \rho \, x \, z \, dv$$

The product is positive when the principal X axis tilts down (looking forward).

(8) Calculator buttons. Click one of these buttons to calculate `Ixx`, `Iyy`, or `Izz` from the radii of gyration Rx (11), `Ry` (12), or `Rz` (13). Sometimes measured values for `Ixx`, `Iyy`, and `Izz` are not available but `Rx`, `Ry`, and `Rz` can be estimated. In this case, each inertia parameter can be estimated by combining the radius of gyration with the total mass:

$$I = M*R^2$$

(9) Roll Radius of Gyration (not a parameter). Use if `Ixx` is not available and `Rx` can be estimated. If used, a guess for Rx might be a third of the vehicle width.

(10) Pitch Radius of Gyration (not a parameter). Use if `Iyy` is not available and `Ry` can be estimated. If used, a guess for Ry might be half the wheelbase.

(11) Yaw Radius of Gyration (not a parameter). Use if `Izz` is not available and `Rz` can be estimated. If used, a guess for Rz might be half the wheelbase.

(12) Link to the library **Suspension Systems: Independent** to use an independent suspension model with CarSim data format.

(13) Link to the screen for the tire properties.

(14) Steering gear ratio (keyword = `RSW`). This is the ratio defines as the steering wheel angle divided by the road wheel angle.

(15) Steering system compliance (keyword = CSMZF). This is the compliance in the steering column and other places not accounted for by the suspension compliance. It is calculated by measuring the steer at a road wheel due to applied steering torque, with the steering wheel fixed. The effect due to suspension compliance is subtracted from that measurement.

(16) Brake gains (keywords = `RMYBK(1)`, `RMYBK(2)`). These are the ratios between brake torque (N-m) and brake input (MPa).

(17) Driveline gains (keywords = `RMYTH(1)`, `RMYTH(2)`). These are the ratios between drive torque (N-m) and dimensionless throttle input.

(18) Link to an animator group data set. The animator group should be set to provide a wire-frame matched to the vehicle.

Location in CarSimEd

CarSimEd Startup
  Runs
    Vehicles: Car

File Location

`Vehicles\Cars_Ed\Cars_Ed.tbk`

10. Advanced Topics

This chapter explains how to perform tasks in CarSimEd that might be helpful after you have some experience. The descriptions in this chapter are similar to those in Chapter 4 — each section is fairly compact, covering just the essence of how to accomplish a task.

Adding a New SIMULINK Model

CarSimEd is installed with a SIMULINK CMEX plug-in DLL file that has the same equations of motion as in the stand-alone solver (EXE) program. It also comes with several example SIMULINK models files (MDL) that make use of the DLL.

As you work with CarSimEd in SIMULINK, you will no doubt create new SIMULINK models with new controller designs or output options. The CarSimEd database accommodates an almost unlimited number of models.

Creating a New SIMULINK Model

The integration between CarSimEd and SIMULINK is done with software contained in the folder `Matlab`, which in turn contains one folder for each SIMULINK model. Each SIMULINK model folder contains a shortcut to the MATLAB program, a CarSimEd CMEX solver module (DLL file), and some extra support files.

To make a new model:

1. Choose among the existing SIMULINK models for the one most similar to the envisioned new model.
2. Duplicate the folder containing the model. For example, if you want to make a new model with the basic model, duplicate the folder `Matlab\cmx_mdl`.
3. (Optional.) Rename the MDL file. If you do this, you must also open the text file `Startup.m` and change the name of the MDL file to match the new name.
4. Edit the MDL file as desired. (To open it, double click the `Matlab` PIF in the new folder.)

Linking to the New Model

1. Use the pull-down menu (3) (see Figure 10.3) to choose the option: **Add**. You will then be prompted to name the new simulation type and to locate an associated `Matlab` PIF for the current vehicle type. (See the next subsection about accessing PIFs from the Windows file dialog box.)

> Notes: This step must be repeated for each vehicle type, to allow CarSimEd to make all of the entries into the internal table.

> The new program will not be used until you select the new simulation type for a new run.

Accessing PIF Shortcut Files

Early versions of Windows used Program Information Files (PIFs) to provide shortcuts to programs. They appear in file browsers as files with the extension PIF. They have largely been replaced by shortcuts, seen in DOS as files with LNK extensions. ToolBook, the program that runs the SGUI part of CarSimEd, does not recognize LNK files but does recognize PIFs. Therefore, PIFs are used to access MATLAB.

When you use the pull-down menu (3) to choose the option: **Add** or **Find Solver For** *Type*, the Windows file dialog box comes up for selecting EXE file. Change the extension in the dialog box from `*.EXE` to `*.PIF` in order to see the PIFs, and then you can link to one of them.

Installing CarSimEd in a New Directory

You can move CarSimEd to a new place in your file system, so long as you do not modify the structure within the CarSimEd directory. For example, you can move the root CarSimEd folder from drive C to drive D. Or, you might copy CarSimEd from a network server to a personal computer. If you move the CarSimEd files after the initial installation, you must make two additional changes:

1. Inform Windows of the new location, in order to start CarSimEd using the runtime ToolBook software.
2. Cause CarSimEd to regenerate text files used to communicate with the plotter, animator, and solver programs.

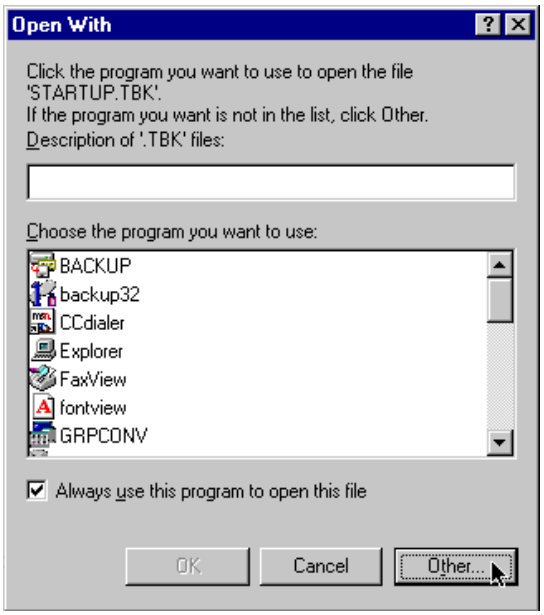
Runtime ToolBook

The CarSimEd database is managed by a program called `Tb40run.exe`, located in the `Tb40` folder in CarSimEd. As part of the CarSimEd installation, the `Tb40run` program is registered with the file extension TBK. If you move the folder containing `Tb40run`, then you must inform Windows of the new locations.

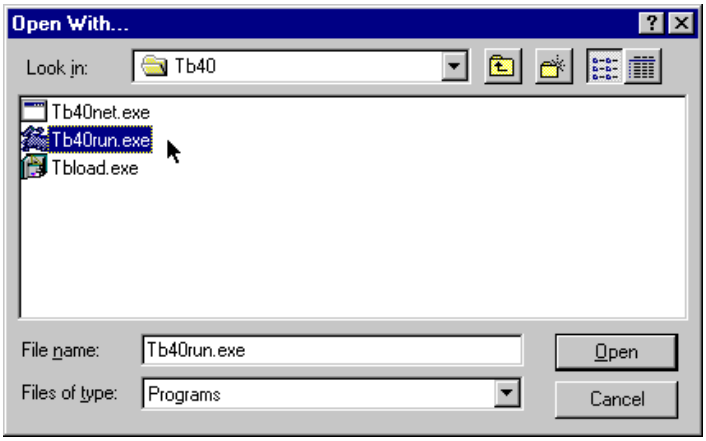
You must associate the file type TBK with the file `Tb40run.exe`. If you know how to do this, go ahead and do it. If not, the following steps describe a method that will work.

1. From the desktop or Windows Explorer, find the file `Startup.tbk` in the CarSimEd folder. Double-click the file to try to open it. If CarSimEd launches, congratulations! No work is needed.

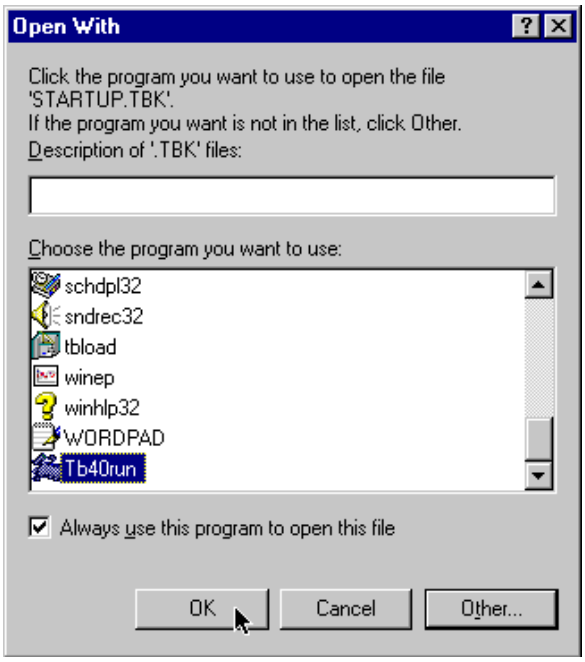
However, more likely, Windows will not know how to open the file, and will bring up a dialog box that looks something like this:



2. Click the button **Other**. This brings up the next dialog box.



3. Locate the file `TB40run.exe`, in the folder Tb40 in CarSimEd. After selecting the file, click the **Open** button to complete the selection and close the dialog box. The previous dialog box should now show the program.



4. Make sure the box "Always use this program to open this file" is checked.

5. Click the **OK** button to close the dialog box.

   You have now associated the file type TBK with the program Tb40run in its new location.

Update Pathnames in PAR Files

The solver programs require absolute pathnames, and CarSimEd creates them "behind the scenes" using PAR text files. When CarSimEd is installed, it "looks" at its location and generates many files using that location to create pathnames. For example, if it is in folder `c:\mycar`, then all pathnames are automatically set to begin with "`c:\mycar`". If you move the CarSimEd directory, you must instruct CarSimEd to regenerate all of the absolute pathnames in the PAR files.

After you have moved the root CarSimEd folder to a new location, to regenerate the absolute pathnames, do the following:

1. Use the **GO** button in the ribbon bar to go to the CarSimEd startup screen (`Startup.tbk`).

2. Go to the data set `Install`.

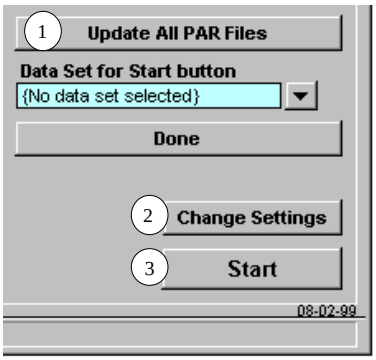


Figure 10.1. Lower-right corner of Startup screen.

3. Click the **Change Settings** button (2). This will hide the CarSimEd logo and reveal more buttons, including those shown in Figure 10.1.

4. Click the **Update All PAR Files** button (1).

> Note: The process of updating all PAR files to generate new absolute pathnames can take several minutes on some computers. The time required depends on the speed of your computer and the number of data sets you have in the CarSimEd database.

5. Click the **Start** button to resume your work.

Importing Data from Another Copy of CarSimEd

There are several occasions where you may want to import data from another copy of CarSimEd. If you obtain an update for CarSimEd, this is how you transfer data entered in the older version. Another reason might be that CarSimEd is installed on several computers, and you want to import data from one installation into the other.

To import data, both installations of CarSimEd must be accessible from the same computer.

1. Go to the library in CarSimEd from which you want to receive the imported data. Any data screen in that library will do.

2. Go to the **File** menu and select the item **Import Data from Other Library**. CarSimEd will bring up a file browser dialog box.

3. Select a library TBK file of the same type as the current one. For example, if the current library contains tire data, you should open only another tire library file. Often, the two files have the same name but are in different folders.

4. Click OK. CarSimEd will copy each data set of the old library and paste the contents into the new one. If a data set already exists with the same name, the copy is given a unique name by appending a number.

> Notes This method is 100% reliable for libraries with only yellow data fields. For libraries with blue links, the names of the linked data sets are copied. However, if the data sets do not exist in the new CarSimEd installation, then the links are not valid and must be reset by hand.
>
> If you are importing data from several libraries, you should import from the "low level" libraries first, to avoid invalid links when importing from "high level" libraries. For example, when importing animation data, import reference frames and shapes before importing groups or camera settings. This is because the groups and camera settings are linked to frames and shapes.
>
> If you select more than 10 data sets to import, a warning message appears warning of possible problems. Due to a low-level interaction between ToolBook and Windows, ToolBook will sometimes crash while trying to import many data sets. It is recommended that you import less than 10 data sets at a time, and that you back up the library file that is the target of the import process.

5. After importing the data, go to the startup screen and update the PAR files. The instructions (page 221) can be summarized as:

    a. Use the **GO** button to go to the startup screen (`Startup.tbk`).

    b. Go to the data set `Install`.

    c. Click the **Change Settings** button.

    d. Click the **Update All PAR Files** button.

    e. Click the **START** button.

Exporting Data to Other Plotting and Analysis Software

CarSimEd is designed for rapid viewing of simulation results using the installed plotting and animation software. However, the software also has provisions for exporting simulation results to other software packages for analysis and viewing. Two methods are described below.

Creating Text Output Files from the Solver programs

By default the output files generated by the solver programs store numerical values in binary format. This speeds up reading and writing and also conserves disk space. Alternatively, the output data can be generated into text files. This file contains all output variables and can be imported into other software.

Start from the **Runs** screen.

1. Go down to the linked **Computation Parameters** data set.
2. Click the **New** button to create a new data set.
3. Change the output file format to **Text** or **ERDText** (use the adjacent button to access a pull-down menu).
   - The **Text** option will cause the simulation to create a simple text output file with a 1-line header followed by columns of numbers. The items on each line will be separated by commas. These files can be imported into spreadsheets and analysis programs that allow comma-delimited columns. However, they cannot be plotted or animated using the built-in CarSimEd tools.
   - The **ERDText** option will cause the simulation to create a text output file with a detailed header followed by columns of numbers. The header has the same information as in the default configuration with binary files. With this option, the installed plotter and animator work the same as with the binary option. The text portion of the file can be imported into other programs, although it might be necessary to delete the header lines at the start of the file.
4. Return to the **Runs** screen by clicking the **Back** button.
5. Select the new **Computation Parameters** data set before making a run to generate text output files. The name of the output file is the ID number shown in the upper-right corner of the **Runs** screen, with the extension ERD (e.g., 231.ERD). The default location of the output files is `CarSimEd\Runs`.

Creating Text Output Files from within the Plotter

The WinEP plotter has the capability of displaying data in text form. This capability can be used to export a subset of variables to other software.

Start from the **Runs** screen.

1. Go to a run data set that involves output variables to be exported.
2. Choose a plot setup involving variables of interest.
3. Click the **Plot** button to launch the WinEP program.
4. Select a plot window in WinEP involving variables of interest.
5. Select the menu item **Show Numbers** under the **View** menu. This displays a dialog box called **Show Numbers** with a scrollable list with numbers separated by spaces.

> Note: Sometimes, due to memory limitations, only some of the values will appear in the list. When this happens, use the **Save Data** button described in the next paragraph.

6. Transfer the numbers:

a. Copy the contents of the field in the dialog box to the clipboard. Then go to the other software and paste the numbers into a field or spreadsheet.

b. Or, click the button **Save Data**. WinEP will prompt for a file name for a text file with the contents of the field in the dialog box. The text file can later be read into other software.

Changing Plot Formatting

Plot appearance is controlled by a format data set. You use the formatting to assign the colors of lines, the size of text, whether a grid is shown, etc.

Each plot description is linked to a format data set (see (7) in Figure 9.7 on page 181). As shipped, every plot description is linked to a format data set with the title `Default Settings`. If you change the default settings, all future plots will have a new appearance.

1. Go to the **Plot Setup: Single** library by clicking the plot setup icon in the ribbon bar () or use the **Plot Setup** command from the **Tools** menu.

2. Go to the **Plot Format** library by following the link (7) (see Figure 9.7 on page 181).

> Note: Another way to get to the library is through the **GO** menu. Look for the menu item `Plot\Format\Format.tbk`.

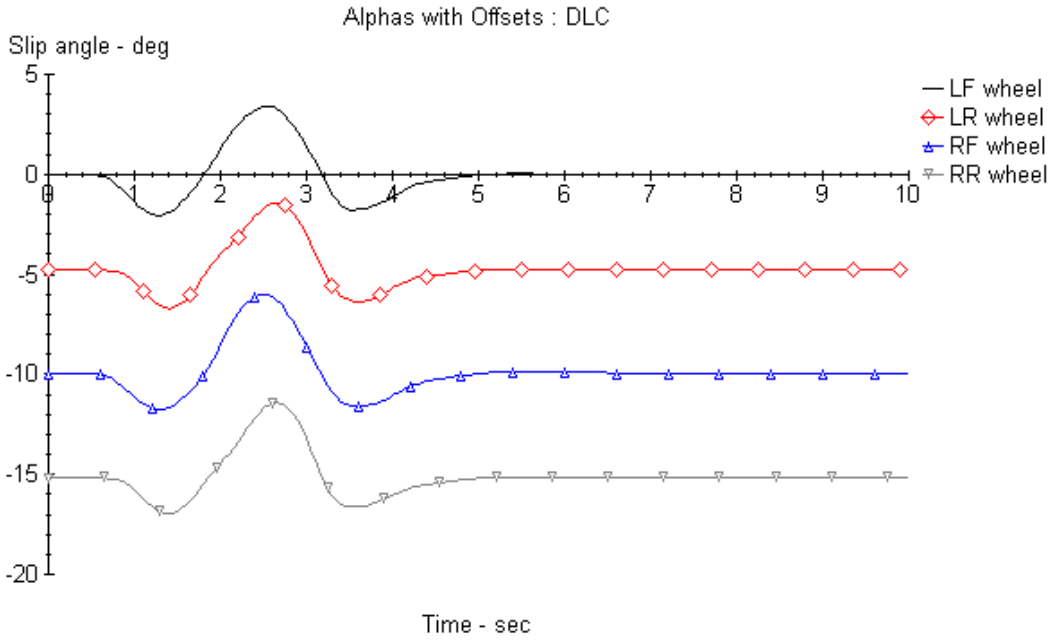
3. Refer to the description of the **Plot Format** screen in Chapter 8 for information about how you can control the plot appearance from the these settings.

4. Modify the data set whose title is `Default Settings`.

> Note: By doing this, all subsequent plots will be made using these modified settings. You can make a new data set, but it will only be used if you set links to it from the **Plot Setup** screen.

Creating Offset Plots

When comparing many similar variables, it is sometimes convenient to offset the plots vertically. For example, the figure below shows four plots that are offset by 5 units.



Making One Plot With Offsets

1. Make a plot of the variables of interest in WinEP. Make sure the window with the plot of interest is the active one by clicking on it. (Click on it even if it is highlighted.)
2. Use the **Data** menu command **Offsets** to bring up the **Offsets** dialog box.
3. Enter values for the offsets in the dialog box.
4. Click the **OK** button to view the revised plot.

Routinely Making Plots With Offsets

1. Go to the **Plot Setup** library by clicking the plot setup icon in the ribbon bar ( ) or use the **Plot Setup** command from the **Tools** menu.
2. Go to the **Data Transforms** library by following the link (see (9) in Figure 9.7 on page 181).

> Note: Another way to get to the library is through the **GO** menu. Look for the menu item `Plot\Transfrm\Transfrm.tbk`.

3. Refer to the description of the **Plot: Data Transforms** screen in Chapter 9 for information about how you can subtract constant values from each data set to apply offsets. Create a new data set with the desired offset values.

4. Go back to the **Plot Setup: Single** screen and go to the plot setup for the plot in which you want to apply the offsets.
5. Set the **Data Transforms** link (12) to the newly created screen.
6. Go to the **Runs** screen and make plots using this modified **Plot Setup** data set.

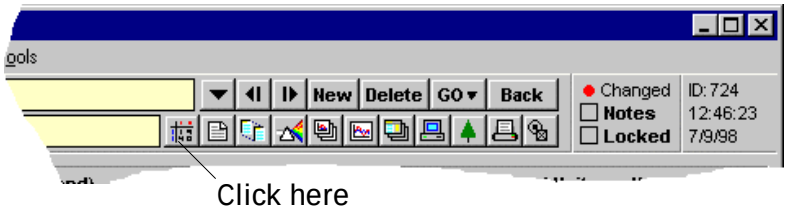
Re-Scaling Tabular Data

You may have occasion to import measured tabular data into CarSimEd. If the units used for the data do not match the requirements for the CarSimEd data screen, you can use a built-in calculator tool to convert the units.

You might also want to take an existing table of number and re-scale it. For example, to make a tire 20% stiffer.

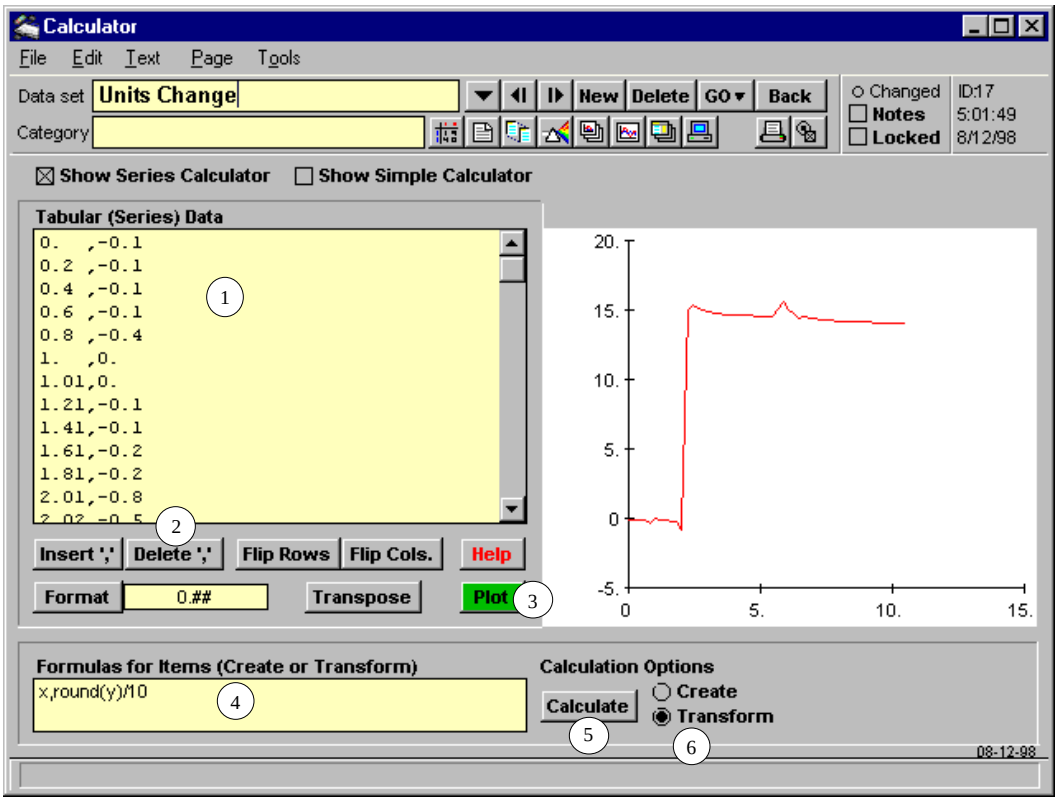
This process is as follows.

1. Copy the tabular data to the clipboard.
2. From any CarSimEd data screen, click the calculator tool button in the ribbon bar or select the **Calculator** command from the **Tools** menu.



3. Paste the tabular data into the tabular data field (1) of the calculator data screen.
4. Select the **Transform** button (6) to configure the calculator for transforming an existing series of numbers (as opposed to creating a new series of numbers).
5. Type definitions for the transformed X and Y values into the definitions field (4). The definitions should have the form: A*X, B*Y where A and B are numerical scale factors. For example, suppose the X values are in inches and Y values are in pounds. To convert to a table where the X values are in millimeters and the Y values are in Newtons, use the definition: 25.4*X, 4.4482216152605*Y.
6. Click the **Calculate** button (5) to perform the transformations.
7. Click the **Plot** button (3) to confirm that the conversion was performed as you expect.
8. Optional. Click the button **Remove ‘,’** (2) if the numbers should not be separated with commas (e.g., for animator shape data).
9. Select the entire contents of the tabular data field (1). The menu command **Select All** under the **Edit** menu (Ctrl+A) is helpful for doing this.

10. Select the **Copy** command from the **Edit** menu (Ctrl+C).



11. Go to the data screen where the tabular data will be stored.

12. If a data set does not exist for the transformed data, create one using the **New** button and give it a name.

13. Paste the clipboard contents into the field used to store the numbers.

See Chapter 9 for details on using the calculator data screen.

Improving the Appearance of the CarSimEd Screens

Windows has many settings for screen size, number of colors, and default font size.

Font Size

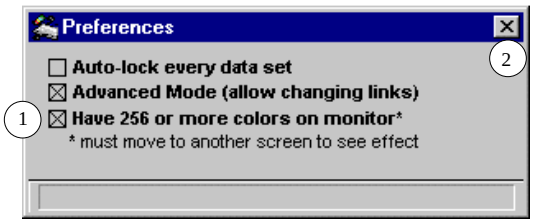
The CarSimEd data screens look best when the system display properties are set for `Small Fonts`. However, the screens should be readable for any valid setting.

Colors

The CarSimEd screens look best when 256 or more colors are supported. If your machine is set to support 16 or fewer colors, CarSimEd has a setting to produce yellow and blue

fields using dot patterns rather than solid colors. A global setting is used to control whether CarSimEd uses solid colors or patterns.

1. From any CarSimEd data screen, click the button in the ribbon bar or select the **Preferences** command from the **Tools** menu. This brings up a dialog box with a few check boxes.


Preferences
Auto-lock every data set
Advanced Mode (allow changing links)
Have 256 or more colors on monitor*
* must move to another screen to see effect
1
2

2. If your machine is set up to support 256 or more colors, check the box **Have 256 or more colors on monitor** (1). Otherwise, un-check this box.

3. Close the dialog box by clicking the X (2) in the upper-right corner.

Size of CarSimEd Data Screens

The CarSimEd data screens are designed to fit on a VGA display (640 x 480).

1. The upper-right corner of the window has three Windows-standard buttons: . The middle button "maximizes" the window to fill the monitor screen (see Figure 10.2). If your monitor has a larger display area than VGA, the result is that the menu bar is put at the top of the total screen, the main part is centered, and the space around the main part is filled with gray.

> Note: In the maximized display mode, pull-down menus associated with triangle buttons do not appear in the correct locations when the buttons are pressed.

2. To return to the normal CarSimEd display, press the F11 key or use the **Size to Page** command from the **Page** menu.

> Note: If you have modified the window size by dragging the window borders, the F11 key (or **Page** menu) is the only way to restore the size.

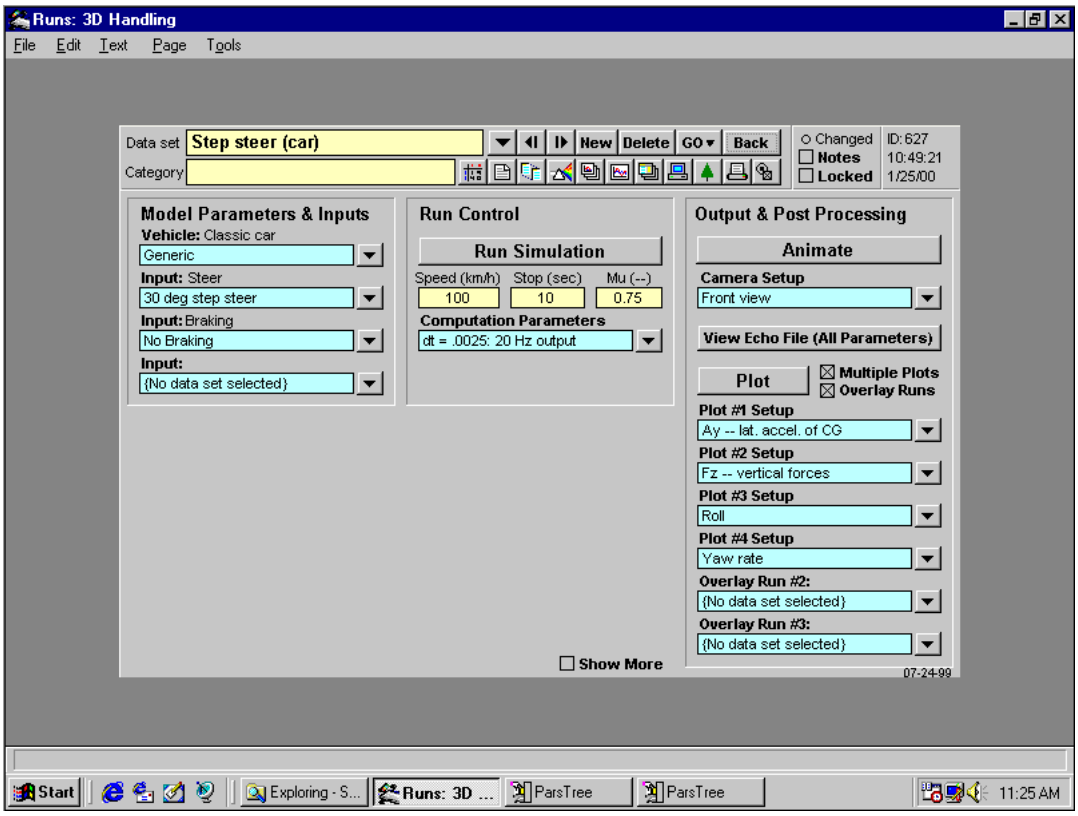


Figure 10.2. Appearance of a data screen that has been maximized.

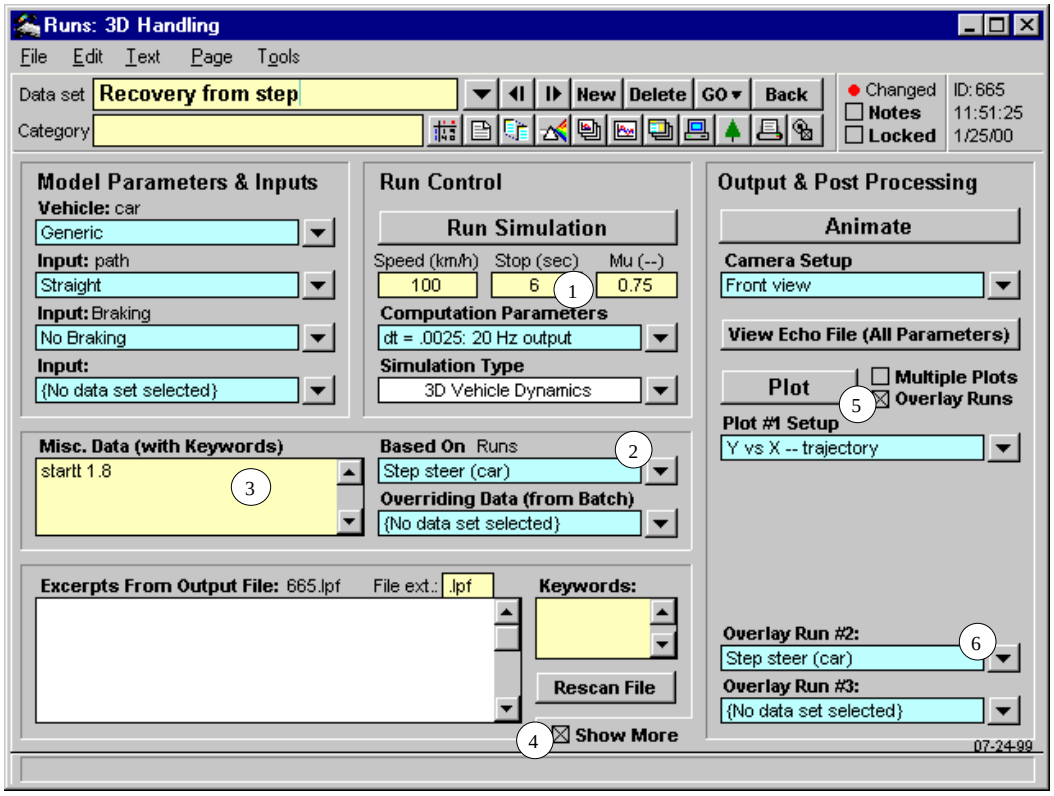
Continuing a Run

The solver programs in CarSimEd are capable of starting a new run right where a previous one left off. However, in the new run, you are free to change anything you want. Some applications of this are:

- You can simulate a drastic change in the vehicle properties, such as a brake failure, tire blow-out, etc.
- You can change controller strategies. For example, specify a hard step-steer input for a second and then resume the run with the driver model trying to bring the vehicle under control.
- You can reproduce an existing run, changing just a few parameters of interest to rapidly investigate vehicle sensitivity to any of the modeled properties.

Note: It is helpful to know which settings on the **Runs** screen have priorities over others. See Chapter 9 section **Runs** for a description of how priorities are determined.

The way to continue a run is to use the **Based On** link of the **Runs** screen to select an existing run.



1. Go to the **Runs** data set for the run you want to continue.
2. Click the button **New** to copy the data set.
3. Make sure that the **Show More** (4) box is checked.
4. Use the **Based On** link (2) to select the run that will be continued.
5. (Optional) To make the time scale for the new run start where the previous run left off, type `startt time` where *`time`* is the stop time from the first run, shown in the Stop field (1) (e.g., it was 1.8 in the original run **Step Steer (car)**). If you do this, change the stop time (parameter name `stopt`) to a larger value (e.g., in the figure it is 6). (The simulation runs from `startt` to `stopt`.)
6. Change any inputs that you want to be different for the new run.
7. Make the new run.
8. (Optional) You can set up overlay plots to show both runs.
    a. Check the **Overlay Runs** box (5).
    b. Select the first turn for the **Overlay Run #2** link (6).

If you made the start time for the new run coincide with the stop time for the first, all plots will show the entire time histories covered in both runs.

> Note: Some of the subsystems perform initializations that prevent perfect continuity. However, the continuity is close enough for most purposes.

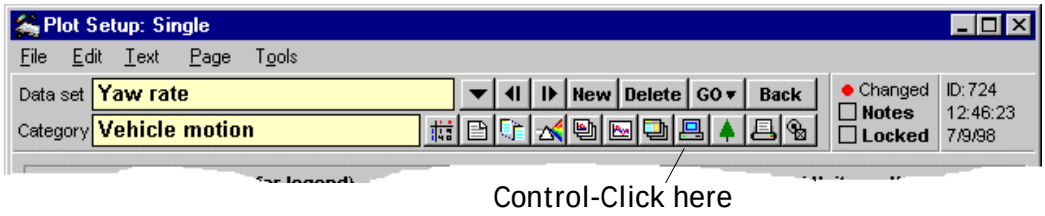
Changing the Default Runs Library

CarSimEd is shipped with four **Runs** libraries: three for using stand-alone (EXE) solver programs, and one for working with SIMULINK. There are two built-in shortcuts to jump directly to a default **Runs** library. One is the **Tools** menu item **Runs**, and the other is the button in the top region of every data screen.

As shipped, the menu item and the button are linked to the **Runs** library used for running the stand-alone 3D car solver program. If you primarily run CarSimEd with SIMULINK, you might wish to change the default link to go to the other **Runs** library.

To assign a new **Runs** library, start with any data screen in CarSimEd.

1. Control-click on the computer icon in the ribbon bar. This brings up a find file dialog box.



2. Select the desired **Runs** library. (As installed, the four options are:

1. `Runs_2d\Runs_2d.tbk`
2. `Runs_3d\Runs_3d.tbk`
3. `Runs_cmx\Runs_cmx.tbk.`
4. `Runs_sus\Runs_sus.tbk.`

Changing a Solver Program

You will probably never have occasion to add new programs to CarSimEd. However, if you use the AutoSim code generator, or are working with someone who is making custom versions of the CarSimEd models, you might be given a new solver program. This section explains how to install it.

CarSimEd supports multiple solver programs. When you click the **Run Simulation** button on the **Runs** screen, CarSimEd searches an internal table for an entry containing the current type of vehicle and the current type of simulation.

> Note: To see the type of simulation, you must check the **Show More** box at the bottom of the **Runs** screen.

For example, in Figure 10.3, the vehicle type displayed above the **Vehicle** blue link (1) is `car` and the **Simulation Type** (3) is `3D Vehicle Dynamics`. The table contains a corresponding pathname (e.g., `Programs\3d_car.exe`) for a solver program that will be run when the **Run Simulation** button (2) is clicked.

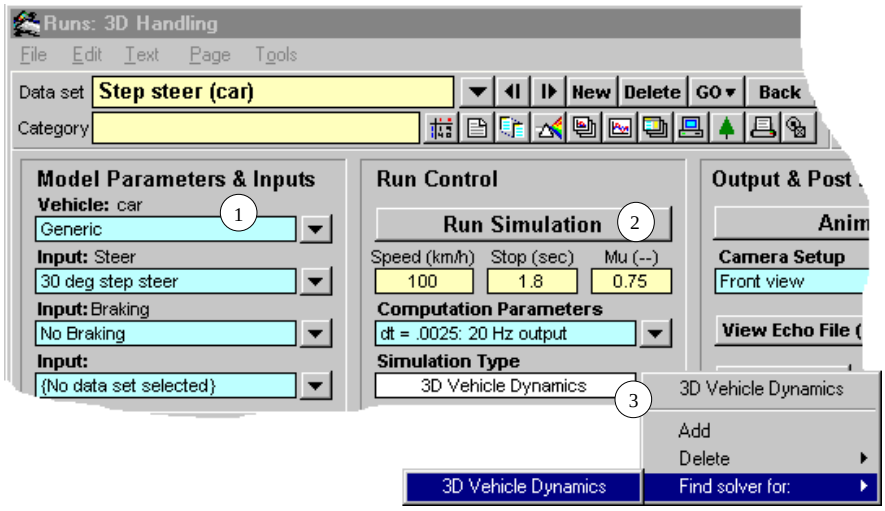


Figure 10.3. Link to a solver program.

> Note: The utility functions on the **Simulation Type** menu (3) are visible only if you have specified **Advanced Mode** in the CarSimEd preferences. To access the preferences, use the **Tools** menu item **Preferences** or click the button in the ribbon bar.

Replacing an Existing Program With Another Program

If the New Program Has the Same Name As the Old Program

1. Replace the old EXE file with a new one.

> Note: As installed, all stand-alone (EXE) solver programs reside in the folder `Programs`.

2. Make a run to confirm that the new program is used. If so, you're done. If not, keep going.

3. Use the pull-down menu (3) to choose the option: **Find Solver For**. *Type* where *Type* is the desired simulation type. In the case of a stand-alone program, you will then be prompted to locate the EXE file associated with the currently select vehicle and simulation types. Locate the new EXE file and select it. Repeat step 2.

If the New Program Has a Different Name Than the Old Program

1. Use the pull-down menu (3) to choose the option: **Find Solver For**. *Type* where *Type* is the desired simulation type. In the case of a new stand-alone program, you will then be prompted to locate the EXE file associated with the currently selected vehicle type. Locate the new EXE file and select it.

2. Make a run to confirm that the new program is used.

Adding An Alternative Program

1. Use the pull-down menu (3) to choose the option: **Add**. You will then be prompted to name the new simulation type and to locate an associated EXE file for the current vehicle type.

> Notes: This step must be repeated for each vehicle type, to allow CarSimEd to make all of the entries into the internal table.
>
> The new program will not be used until you select the new simulation type for a new run.

Removing a Simulation Type

If you want to delete a simulation type from the menu **Simulation Type** (3), then:

1. Use the pull-down menu (3) to choose the option: **Delete**. along with the simulation type that you want removed.

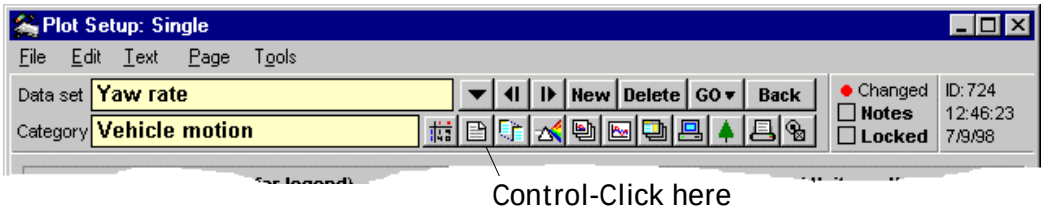
> Note: CarSimEd will not delete any EXE files. The **Delete** menu item only removes an item from the menu.

Changing the Default Text Editor

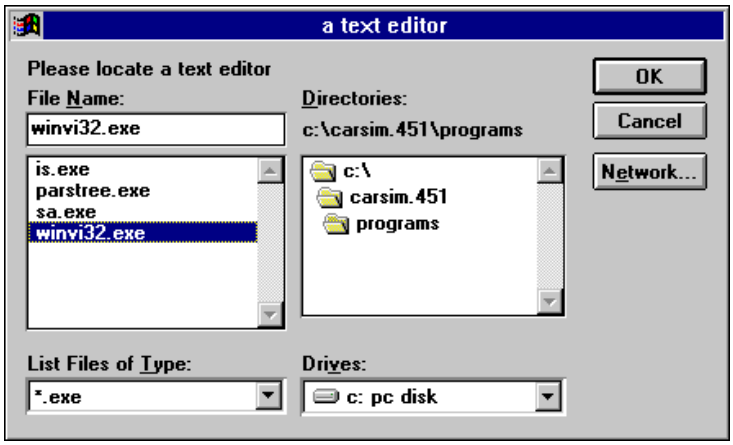
The **View Echo File (All Parameters)** button on the **Runs** screen automatically loads a text file into an editor. As installed, the default text editor is a program called `WinVI`. You can replace `WinVI` with a different Windows editor if you prefer.

To assign a new editor, start with any data screen in CarSimEd.

1. Control-click on the text icon in the ribbon bar.



This should bring up the following dialog box.



2. Link to a text editor using the above dialog box.
3. Click the **OK** button after you have selected the desired text editor.

11.Trouble Shooting

This chapter lists some problems that can occur with CarSimEd, along with suggested solutions.

CarSimEd is an integrated collection of programs: solver programs, the database, the animator, the plotter, and a few utility programs. The potential warnings, error messages, and crashes are organized by the type of program in CarSimEd. All parts of the software are susceptible to problems related to filenames, and therefore this topic is addressed first.

File System Errors

Information is passed between the various parts of the CarSimEd through text files. If CarSimEd attempts to access non-existent files, error messages are generated. However, CarSimEd has a way to automatically update all filenames, making this kind of error reasonably easy to prevent.

Moving CarSimEd to a New Location

When you move CarSimEd to a different folder, pathnames written in hundreds of text file are no longer valid (they reference the old folder name and location). It is also possible that the ToolBook program needed to run the database will not be found.

To avoid errors when you move CarSimEd, be sure to follow the instructions in the Chapter 4 section **Installing CarSimEd in a New Directory**. The key steps in this process are:

1. Go to the CarSimEd startup screen.
2. Click the **Change Settings** button to reveal more buttons.
3. Click the button **Update All PAR Files.**

Changing Names of Data Sets

The same kind of problem can occur when you import data from another copy of CarSimEd, or when you change the name of a data set that is referenced from elsewhere in the database.

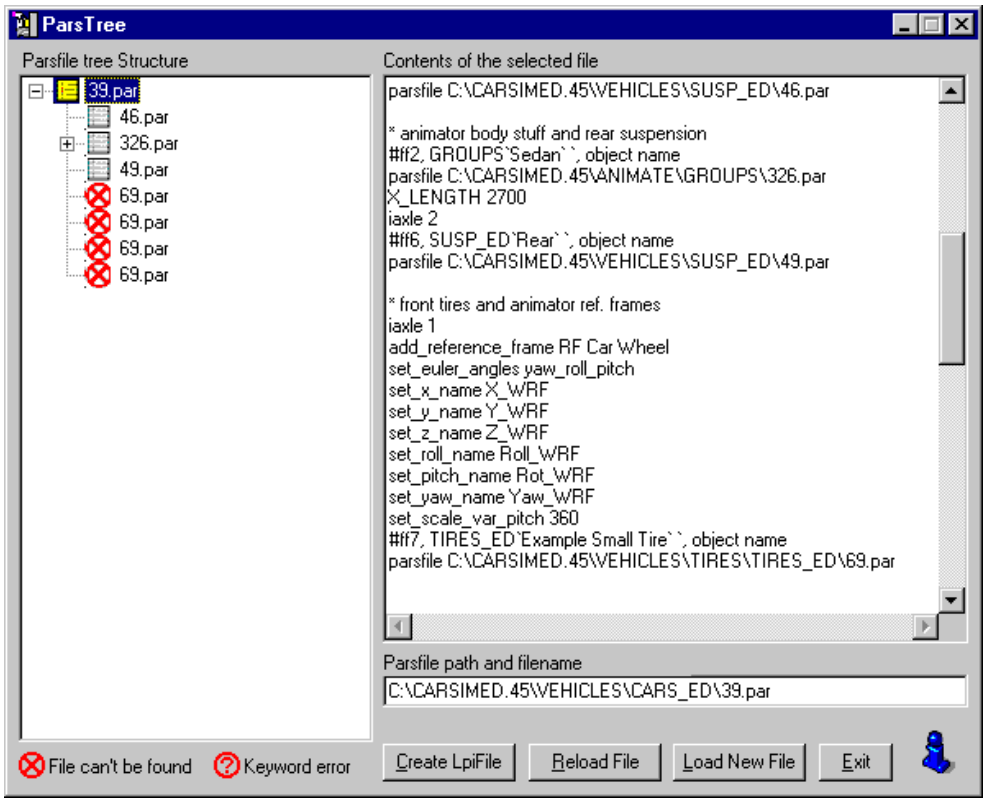
If you click the **Run Simulation** button and the solver stops with an error message that a file can’t be found, then there are two possible solutions.

1. First, go to the startup screen and click the button **Update All PAR Files** as described above.

2. If that doesn't work, there is probably a link to a data set that no longer exists because the original was deleted or renamed.

In the second case, you should be able to identify the library containing the non-existent data set from the error message. You need to figure out which library has the bad link. For example, if the non-existent file is in the folder `Vehicles\Tires\Tires_ed\`, then the problem is probably with a tire data set. Tires are selected from the car and data screen, so that would be the place to look for the bad link.

Bad links can be quickly located by using the ParsTree program in CarSimEd. To run it, click on the button with the tree icon: . Alternatively, go to the **Tools** menu and select the item **View Parstree**. The ParsTree program brings up a window with two panes, similar in format to the Windows Explorer. Bad links are indicated with the symbol . For example, the figure below shows the ParsTree display for a vehicle PAR file with a bad link for the tire (repeated twice).



Database (ToolBook) Problems

The database in CarSimEd is run under a program called ToolBook, from Asymetrix. The version being used in CarSimEd includes 16-bit library routines which can crash.

Most users have never experienced a crash of ToolBook. The problem is not reproducible, but it does exist.

Whenever you switch from one ToolBook library to another one, the file is automatically saved. Therefore if a crash occurs you will not lose any data from libraries other than the one that is open.

Risky Operations

Here are the operations that appear to carry some risk.

- Importing data sets into a library from another CarSimEd library can crash ToolBook. The problem has something to do with memory management within ToolBook. To reduce the risk, do not import more than 10 data sets at a time, and save the ToolBook file after each one.
- Having two copies of ToolBook running is risky and should be avoided. If you launch CarSimEd and it is already running, you will be given a warning and the option to Quit the second instance. We recommend you use this option.
- CarSimEd includes a Find option in ToolBook. It can lead to internal ToolBook conflicts and should not be used casually.

Recovery From a Crash

If ToolBook crashes, it usually disables the part of Windows that supports 16-bit code, requiring that Windows be restarted. Usually this is all that is needed.

Sometimes, however, when you restart CarSimEd you will get a message that the file `Startup.tbk` has been corrupted. CarSimEd includes a backup copy of the `Startup.tbk` file, named `Startup_.tbk`. If the main one gets corrupted, the simplest fix is to do the following:

1. delete the old (corrupt) file,
2. duplicate `Startup_.tbk`, and
3. rename the copy as `Startup.tbk`.

If you made new data sets in your Startup library, you can import the data sets. The fix is:

1. duplicate `Startup_.tbk`,
2. open the duplicate file in ToolBook (double click its icon),
3. import the new data sets from the old (corrupt) `Startup.tbk` file,
4. exit ToolBook,
5. delete the old (corrupt) file, and
6. rename the copy as `Startup.tbk`.

If you lose the backup in the CarSimEd folder, you can copy the `Startup.tbk` file from the CarSimEd CD. Windows will give the “Read Only” file permission because it comes from a CD. You should change this to Read/Write permission before trying to use CarSimEd.

Solver Programs

Once a run starts, the solver programs have three methods of stopping.

1. A normal stop, in which the simulated test is completed. In this case the solver program quits at the end of the run, returning you to the **Runs** screen where you can click a button to view an animation or inspect plots. (There are several parameters that are checked to see if the run should stop: the stop time, a minimum speed threshold, a maximum roll angle, and a maximum distance traveled.)

2. A controlled stop due to an input error. In this case the window for the solver program stays visible, with an error message displayed. This is usually due to a non-existent file name, and is corrected by going to the **Startup** screen and updating the text files, as described on page 236. If this does not solve the problem, locate and open the LOG file, described in Chapter 5, and use it to determine how many input files were read, and which file caused the problem.

3. An uncontrolled stop. In this case a Windows system-level error message is generated. The problem is usually numerical, causing an excessively large number to be passed to the CPU. The problem is usually due to bad input parameter values that cause the system to become unstable. To help track down the problem:

    a. View the echo file made prior to the start of the run. The echo file has the extension LPO and is described in Chapter 5. Look over the parameters for something that is not right, like a very low mass, negative spring rate, etc.

    b. Repeat the run using a smaller time step.

    c. If the run made some progress before crashing, use the animator and plotter to try to figure out what part of the model misbehaves first.

Plotter and Wire-Frame Animator

The plotter and animator viewer programs do not create data nor do they open data files with write permission. Neither is likely to corrupt any of your data. The worst they can do is crash. The potential problems fall into two categories: the file system and numerical errors.

File System Problems

The animator reads the same PAR files as the solver programs, plus it reads the ERD files created by the solver programs. The plotter reads a similar file (with extension PLT) and also the ERD files.

If any of these files have been deleted or renamed, either viewer will show an error message and quit. Here is a quick check list:

- Has the run been made? If a run has not yet been made, then there is no ERD file to read. The viewer will show an error message naming the missing ERD file.
- Are the links from the **Runs** screen valid, and all links “down the tree?” If not, either viewer will show an error message naming the missing PAR file. Try to locate the problem using the Parstree program as described earlier on page 236.
- Was the output simulation file specified to be plain text? If so, neither viewer will work right. Check the **Computation Parameters** data set used for the run. If the output file format is set to **Text**, then the viewers will not work. (Make the run using one of the other output file formats: **Binary** or **ERD Text**.)

Numerical Problems

The animator performs 3D coordinate transformations using data from the shape files and from the simulation output file. Occasionally it will try to divide by a zero or otherwise perform an illegal numerical operation. Here are two possibilities:

- Was the simulated test stable? It is possible for a solver program to write invalid numbers into the output file. When either viewer reads those numbers, it crashes. To check for this, see if the results from the run in question cause problems with both the animator and plotter. If so, the problem is really with the solver program and is probably due to bad model parameters or a time step that is too big.
- For the animator, have you moved the camera point inside the vehicle? Usually this is safe, but it will occasionally create a singularity in the 3D transformations (Divide By Zero) and cause the animator to crash. If this happens while changing the camera location interactively, you can simply re-start the animator. If it crashes right away, try choosing a different camera setup from the **Runs** screen.

Appendix A — Glossary

This appendix contains a glossary of words that have specific interpretations in the context of CarSimEd.

Animator — a program in CarSimEd that is used for animating simulation results as wire-frame representations of vehicles.

AutoSim™ — a code generator that was used to create the solver programs in CarSimEd. AutoSim originated at UMTRI, and is now licensed, maintained, and supported by MSC.

Blue field — Blue fields with adjacent triangle buttons are common in CarSimEd. Each represents a link to another data set. The name of the data set is shown in the blue field, and the triangle button is used to display a pull-down menu (see **Link**).

CarSim™ — the commerical version of CarSimEd. CarSim can be licensed in several forms from MSC. CarSim comes with more detailed vehicle dynamics models that include a driveline, 3D ground input, wind, and variable ground friction. Many of the linear coefficients in the CarSimEd models are replaced with nonlinear tables in order to accurately simulate a wider range of conditions.

CarSimEd™ — an integrated software package for automotive vehicle dynamics simulations. CarSimEd (short for CarSim Educational) is free and can be downloaded from the MSC web site: `www.trucksim.com`. Unless otherwise noted, the name CarSimEd refers to the basic package with stand-alone solver programs (EXE files) and MATLAB/SIMULINK DLL plug-ins.

CarSimEd database — the collection of all data libraries within CarSimEd.

CarSim Educational — see CarSimEd.

CMEX — an executable module for MATLAB/SIMULINK that was created by compiling C code. The CMEX versions of the CarSimEd models are contained in DLL (dynamic link library) files.

Data screen — a view of a data set contained in the CarSimEd data base. A data screen is where you enter the parameters that define your target simulation. It is a data set as viewed through a graphical user interface.

Data set — a collection of parameter values and related information, organized for display on a CarSimEd screen. Data sets are contained in library files and edited in data screens.

DLL — dynamic link library. Files with this extension contain library code that is loaded into memory as needed.

Echo file — a text file, produced by a solver program, that lists every input parameter. The echo file documents the conditions that were simulated in a run. All echo files made

by CarSimEd solver programs are written in the PARSFILE format (see Appendix F). Echo files can also be used as inputs to the solver programs for future runs, to restart (continue) a run.

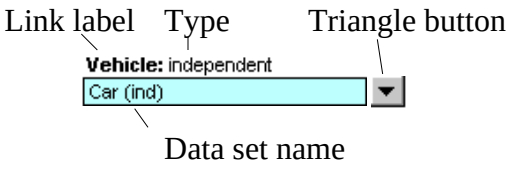
ERD file — a file stored in a standard format that supports automated plotting, animation, and other forms of post-processing. Output files produced by CarSimEd solver programs are written in ERD format (see Appendix C). The plotter and animator programs in CarSimEd read ERD files. ERD stands for Engineering Research Division, the group at UMTRI that originated this format.

Go — in the context of CarSimEd, going to a different data set, screen, or library means having CarSimEd change the screen display to view a different data set. The new data set could be from the same library or a different one. (A **Go** button is in the ribbon bar of every data screen, and is used to transfer to different libraries. The triangle button next to the data set title is used to transfer to different data sets within the same library.)

Keyword — a "word" comprised of letters and other characters that is recognized by a computer program. In CarSimEd, all vehicle parameters are identified in input files with keywords. Keywords in CarSimEd are not case dependent (`speed`, `SPEED`, and `SpeED` all appear the same to the solver programs). Keywords cannot include spaces.

Library — a file containing one or more data sets of the same type, plus a standard graphic interface for viewing these data sets. All CarSimEd library files have the extension TBK.

Link — a connection from one data set to another, indicated with a blue field and an adjacent triangle button. For example, here is a link to a vehicle data set.



The adjacent button has a pull-down menu that can be used to rapidly change the link to a different data set or library. The menu can also be used to "follow the link" and go to the data set.

MATLAB® — a mathematical computation and visualization package available from The MathWorks, Inc. An add-on module called SIMULINK can optionally be used with CarSimEd.

MSC — Mechanical Simulation Corporation. The company that licenses, maintains, and supports CarSim and CarSimEd.

Numerical integration — a computation method used to solve the differential equations that define a mathematical model of vehicle dynamics. Output variables produced by the models in CarSimEd are based on internal variables that are calculated over time using numerical integration of state equations.

Parameter — parameters are quantities that remain constant during a run. Any numbers that you see on a data screen are either scalar or tabular parameters. Variables, on the other hand, are continuously computed by the solver programs and are usually viewed with the plotter and animator.

PARSFILE (parameter file) — a keyword-based text file used to communicate between various CarSimEd programs. (See Appendix F for details about the format.) Most have the DOS extension PAR, but some files have different extensions yet follow the PARSFILE format. (For example, the LPO and LPF files list all parameter values used in a run.)

Ribbon bar — the collection of buttons and user controls displayed at the top of nearly every CarSimEd data screen.

Run — Shorthand for "run a simulated test" (e.g., "make a run"). Also used to refer to the "outcome of a simulated test run" (e.g., "look at a run").

Screen — short for "screen layout associated with a library." This manual has many references to the act of going from one screen to another. "Changing the screen" means taking an action that changes the view to show data in a different library.

SGUI — Simulation Graphical User Interface. The architecture of CarSimEd (data screens, plotter, solver programs, animator) has been used for other software packages. The generic architecture, created at UMTRI and currently developed and maintained by MSC, is called the SGUI.

SIMULINK® —a software package for modeling, simulating, and analyzing dynamical systems in general. It runs under MATLAB®. SIMULINK and MATLAB are available from the MathWorks, Inc. Versions of the CarSimEd models (CMEX DLL files) are provided for users who wish to run in the SIMULINK environment.

Solver program — a program that numerically solves the equations of motion of a vehicle model to simulate a test. The CarSimEd solver programs are customized with the equations of motion for specific vehicle models. They handle all required calculations and input/output.

TBK files — binary files that contain the CarSimEd libraries. In addition, some of the code needed to make the CarSimEd buttons work is stored in TBK files. All necessary TBK files are provided in the CarSimEd installation, and they are managed automatically by CarSimEd. TBK files are native to the ToolBook software.

ToolBook™ — a "Visual Authoring System" from Asymetrix, Inc. The SGUI is largely programmed within ToolBook. The running of ToolBook code and management of ToolBook libraries is done by the ToolBook runtime package, which is included with CarSimEd.

Triangle button — a button with a triangle icon ( ), usually located adjacent to a field. you click on the button using the left mouse button to display a pull-down menu.

TruckSim™ — an integrated software package (similar to CarSimEd) available from MSC for simulating and analyzing the braking and handling behavior of trucks and tractor-trailer combinations.

UMTRI — The University of Michigan Transportation Research Institute. Most of the modeling and simulation technology in CarSimEd are based on knowledge and methods developed at UMTRI.

Variable — an output quantity computed by the CarSimEd vehicle models. In contrast to a parameter, a variable can change with time in a way that must be simulated using a detailed vehicle dynamics model. Output variables can be plotted with WinEP, the plotter embedded in CarSimEd. Some of the output variables are read by the animator embedded in CarSimEd to create an animation of the simulated vehicle motion.

WinEP — the Windows Engineering Plotter is the plotting program in CarSimEd. It makes X-Y plots from data read from ERD files, and is used to view simulation results. It can also be used to view data from other sources, such as test data.

WinVI — a public-domain text editor included with CarSimEd that is used for editing and viewing text files.

Yellow field — a rectangular box for text or numerical information on a data screen that you can edit directly. For example, to change the vehicle wheelbase, find the yellow field, click on the field, and change the value using the mouse and keyboard.

Appendix B — Vehicle Dynamics Terminology

This appendix defines specialized terms applicable to vehicle dynamics solver programs. The definitions draw on two sources:

1. SAE Recommended Practice J670e, **Vehicle Dynamics Terminology** (first issued 1952, last updated 1976).
2. ISO 8855, **Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary** (1991).

This appendix is provided because neither of these two sources fully define the terminology that is now common in the field of vehicle dynamics simulation. SAE J670e was last updated before solver programs existed for complex models. ISO 8855 is more simulation-oriented, but is limited in scope.

> Note: SAE is in the process of replacing J670e with a version that includes more modern definitions. Although the parts of J670 that are related to coordinate systems and basic terminology are complete, it is the practice of SAE to discourage distribution of draft material. Nonetheless, the definitions in this appendix are thought to be compatible with the future version of J670.

This appendix is intended to establish useful conventions for vehicle dynamics simulation, maintaining compatibility with SAE and ISO when practical. Exceptions are noted when they occur. The terminology applies to all simulation software developed and maintained by Mechanical Simulation Corporation (CarSimEd, TruckSim, etc.).

Terms are not defined unless they are necessary for describing vehicle solver programs such as those that are supplied by Mechanical Simulation Corporation. The level of detail is matched to models that are system based, rather than component based. To obtain generality, terms are defined without reference to specific models. Terms that are defined in this document are written in italics, followed by symbols when they exist. Definitions that are taken from SAE J670e and/or ISO 8855 are designated SAE and ISO, respectively, in parentheses. Conflicts with SAE or ISO are described in numbered notes at the end of this appendix.

Vectors and Angles

Vectors

Acceleration vector (linear) — time derivative of the *velocity vector* of a point.

Angular acceleration vector — time derivative of the *angular velocity vector* of a *reference frame*.

Angular velocity vector— vector describing the absolute 3D angular velocity of a *reference frame* with respect to the *inertial reference*. (Formally, it is a quantity that satisfies the equation:

$$\dot{\mathbf{r}} = \omega \times \mathbf{r}$$

where $\mathbf{r}$ is a vector fixed in a reference frame and ω is the angular velocity vector of this reference frame.)

Position vector — *vector* describing the position of one point relative to a reference point. Unless specified otherwise, the reference point is the origin of the *earth-fixed coordinate system*.

Vector — an object that has a direction in 3D space and a magnitude. The existence and meaning of a vector are not dependent on the choice of coordinate or axis system.

Velocity vector — time derivative of the *position vector* of a point.

Angles

An angle implies a rotation **from** a reference line **to** another line, in the plane containing both lines, **about** an axis that is perpendicular to both lines. The direction of the axis defines the sign convention of the angle, based on a right-handed rotation. (Line up your right-hand thumb with the axis direction, and your fingers curl in the direction of a positive rotation.) Sometimes an angle is defined between a plane and a line. In this case, the angle is taken from a projection of the line into the plane, as shown in Figure B.1 for an angle between plane ABC and line AD.

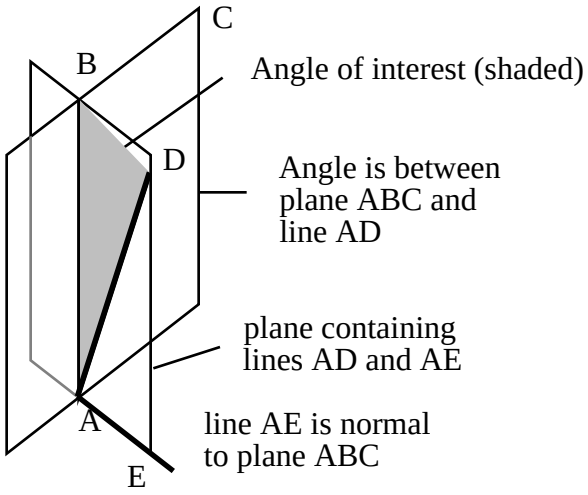


Figure B.1. Angle between a plane and a line.

Resultant Force and Moment Vectors

All actions on a body that would cause it to accelerate in translation or rotation if not opposed by other actions can be combined into a single *resultant force vector* and a single *resultant moment vector* about a point at which the resultant force vector is assumed to apply. The resultant moment vector depends on the location of the point where the resultant force is assumed to apply. Resultant force and moment vectors are used to describe actions on the vehicle due to the ground, the air, and impacts with other objects.

Axis Systems and Coordinate Systems

Discussion

Multibody vehicle dynamics models are typically generated using right-handed *axis systems* and *coordinate systems*. The axis orientation for ISO 8855 has X pointing forward, Z pointing up, and Y pointing to the left-hand side of the vehicle. SAE J670e has Z pointing down, X pointing forward, and Y pointing to the right.

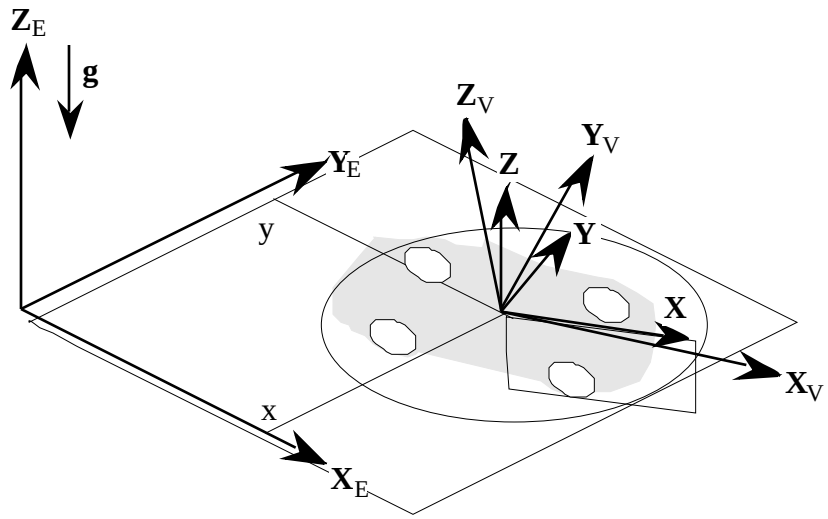
The Z-up convention is used in CarSimEd, TruckSim, and other MSC vehicle models for several reasons:

- Plots of Y vs. X show a top view of vehicle trajectories;
- plots of Z vs. X show a side view of vehicle trajectories;
- vertical tire forces are always positive; and
- wheel spin rates are positive for forward vehicle speeds.

With the Z-down convention, all of these signs are reversed.

Alternative right-handed systems are theoretically compatible with the following definition, so long as X is longitudinal, Y is lateral, and Z is vertical. However, the sign conventions of many variables are dependent on the directions in which the axes are pointing. For example, yaw angle is always a rotation about the Z axis. With Z-up (ISO, MSC), positive yaw implies a left-hand turn; with Z-down (SAE), positive yaw implies a right-hand turn.

Figure B.2 shows the three axis systems associated with the entire vehicle. The *intermediate axis system* is used the most for vehicle-level definitions. It has a Z axis that is parallel to the gravity vector, and an X axis that is in the same vertical plane as the vehicle longitudinal axis. Figure B.3 shows the tire and wheel axis systems.



Notes
1. **Z** is parallel to $\mathbf{Z}_E$
2. **X** is in the vertical plane containing $\mathbf{X}_V$
3. The angle between $\mathbf{X}_E$and **X** is ψ (yaw)

Figure B.2. Earth, vehicle, and intermediate axis systems.

Definitions of Terms

Axis system — a set of three orthogonal X, Y, and Z axes. In a right-handed system, **Z** = **X** × **Y.** (Here, **X**, **Y**, and **Z** refer to an arbitrary orthogonal axis system.)

Coordinate system — a numbering convention used to assign a unique ordered trio of numbers to each point in a *reference frame*. A typical rectangular coordinate system consists of an *axis system* plus an origin point.

Earth-fixed axis system ($\mathbf{X}_E$, $\mathbf{Y}_E$, $\mathbf{Z}_E$) — right-handed orthogonal *axis system* that serves as the global, *inertial reference* frame. The $\mathbf{Z}_E$ direction is parallel to the gravity vector. $\mathbf{Z}_E$ points up in the MSC and ISO systems. (See Figure B.2.) (ISO, SAE)

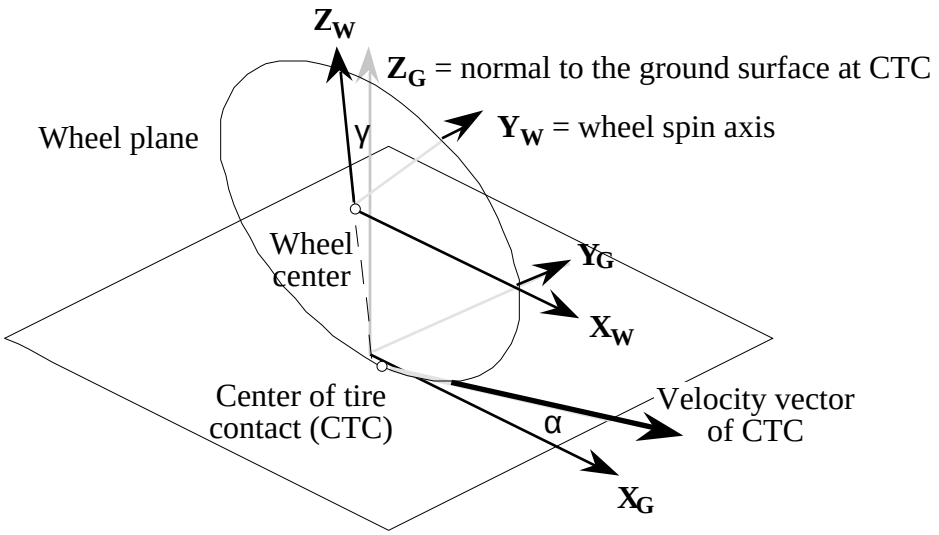


Figure B.3. Tire and wheel axis systems.

Earth-fixed coordinate system — coordinate system based on the *earth-fixed axis system* . Its origin normally lies somewhere on the Earth's surface.

Inertial (Newtonian) reference — a reference frame that is assumed to have zero *acceleration* (both *linear* and *angular*) and zero *angular velocity.*

Intermediate axis system (**X**, **Y**, **Z**) — right-handed orthogonal *axis system* whose **Z** axis is parallel to $\mathbf{Z}_E$, and whose **Y** axis is perpendicular to both $\mathbf{Z}_E$ and $\mathbf{X}_V$. This axis system can be obtained by rotating the Earth-fixed axis system about $\mathbf{Z}_E$ by the vehicle *yaw angle* (ψ). (See Figure B.2.) (ISO)

Reference frame — a geometric environment in which points remain fixed with respect to each other at all times.

Ground axis system ($\mathbf{X}_G$, $\mathbf{Y}_G$, $\mathbf{Z}_G$) — right-handed orthogonal *axis system* whose $\mathbf{Z}_G$ axis is normal to the ground, at the *center of tire contact*, and whose $\mathbf{X}_G$ axis is perpendicular to the wheel *spin axis* ($\mathbf{Y}_W$). For uneven ground, a different ground axis system can exist for each tire.(See Figure B.3.) (SAE, ISO[1])

Ground plane — a reference plane tangent to the ground surface at the tire *contact center*. For uneven ground, a different ground plane can exist for each tire.

Vehicle axis system ($\mathbf{X}_V$, $\mathbf{Y}_V$, $\mathbf{Z}_V$) — right-handed orthogonal *axis system* fixed in the *vehicle reference frame.* The $\mathbf{X}_V$ axis is primarily horizontal in the *vehicle plane of symmetry* and points forward. The $\mathbf{Z}_V$ axis is vertical and the $\mathbf{Y}_V$ axis is lateral. The directions should coincide with the *earth-fixed axis system* when the vehicle is upright and aligned with the $\mathbf{X}_V$ axis parallel to the $\mathbf{X}_E$ axis. (See Figure B.2.) (SAE, ISO)[2]

Vehicle plane of symmetry — the lateral center plane of the vehicle. The $\mathbf{Y}_V$ direction is normal to this plane.

Vehicle reference frame — reference frame associated with the vehicle body. It is typically defined to coincide with the undeformed body of the vehicle body structure.

Wheel axis system ($\mathbf{X}_W$, $\mathbf{Y}_W$, $\mathbf{Z}_W$) — right-handed orthogonal *axis system* whose $\mathbf{Y}_W$ axis is parallel with the *spin axis* of the wheel and whose $\mathbf{X}_W$ axis is perpendicular to $\mathbf{Z}_G$. (See Figure B.3.)

Mathematical Definitions

All of the coordinate systems and axes are built from four reference directions:

1. vertical, as defined by the direction of the gravity vector ($-\mathbf{Z}_E$),
2. the X axis of the *vehicle reference frame* ($\mathbf{X}_V$),
3. the Y (spin) axis of a wheel of interest ($\mathbf{Y}_W$), and
4. the direction normal to the ground at the *center of tire contact* ($\mathbf{Z}_G$).

Table B.1 defines the intermediate, wheel, and ground axis systems in terms of $\mathbf{X}_E$, $\mathbf{Y}_E$ and $\mathbf{Z}_E$, $\mathbf{X}_V$, $\mathbf{Y}_V$ and $\mathbf{Z}_V$, $\mathbf{Y}_W$, and $\mathbf{Z}_G$. Two equivalent definitions are provided for the **X** and **Y** directions (*intermediate axis system*).

Table B.1. Axis system definitions.

Name	X Direction	Y Direction	Z Direction
Earth $\mathbf{X}_E$, $\mathbf{Y}_E$, $\mathbf{Z}_E$	$\mathbf{X}_E$	$\mathbf{Y}_E$	$\mathbf{Z}_E$
Vehicle $\mathbf{X}_V$, $\mathbf{Y}_V$, $\mathbf{Z}_V$	$\mathbf{X}_V$	$\mathbf{Y}_V$	$\mathbf{Z}_V$
Intermediate **X**,**Y**, **Z** ($\mathbf{Z}=\mathbf{Z}_E$)	$\frac{(\mathbf{Z}_E \times \mathbf{X}_V)\times \mathbf{Z}_E}{\lvert(\mathbf{Z}_E \times \mathbf{X}_V)\times \mathbf{Z}_E\rvert}$ or $\mathbf{X}_E \cos(\psi) + \mathbf{Y}_E \sin(\psi)$	$\frac{\mathbf{Z}_E \times \mathbf{X}_V}{\lvert\mathbf{Z}_E \times \mathbf{X}_V\rvert}$ or $\mathbf{Y}_V \cos(\psi) - \mathbf{X}_E \sin(\psi)$	$\mathbf{Z}_E$
Wheel $\mathbf{X}_W$, $\mathbf{Y}_W$, $\mathbf{Z}_W$	$\frac{\mathbf{Y}_W \times \mathbf{Z}_R}{\lvert\mathbf{Y}_W \times \mathbf{Z}_R\rvert}$	$\mathbf{Y}_W$	$\frac{(\mathbf{Y}_W \times \mathbf{Z}_R)\times \mathbf{Y}_W}{\lvert(\mathbf{Y}_W \times \mathbf{Z}_R)\times \mathbf{Y}_W\rvert}$
Ground $\mathbf{X}_G$, $\mathbf{Y}_G$, $\mathbf{Z}_G$ ($\mathbf{X}_G = \mathbf{X}_W$)	$\frac{\mathbf{Y}_W \times \mathbf{Z}_R}{\lvert\mathbf{Y}_W \times \mathbf{Z}_R\rvert}$	$\frac{\mathbf{Z}_R \times (\mathbf{Y}_W \times \mathbf{Z}_R)}{\lvert\mathbf{Z}_R \times (\mathbf{Y}_W \times \mathbf{Z}_R)\rvert}$	$\mathbf{Z}_G$

Note: the axes names $\mathbf{X}_W$, $\mathbf{Y}_W$, and $\mathbf{Z}_W$ have a different meaning in ISO 8855.

Entire vehicle

Size

Wheelbase L_{WB}— the distance between the *centers of tire contact* on one side of the vehicle. Wheelbase usually varies slightly with suspension deflection. (ISO)

Components of vectors

Forces, moments, and motion vectors for the entire vehicle are commonly decomposed into three rotational and three translational terms. The following adjectives should be used.

Lateral — **Y** component of force or translational motion vector. (ISO)

Longitudinal — **X** component of force or translational motion vector. (ISO)

Pitch — **Y** component of moment or rotational motion vector. (ISO)

Roll — **X** component of moment or rotational motion vector. (ISO)

Vertical — **Z** component of force or translational motion vector. (ISO)

Yaw — **Z** component of moment or rotational motion vector. (ISO)

Points

C.G. (Center of gravity) — a point in the *vehicle reference frame* that coincides with the center of mass of the entire vehicle when the suspensions are in equilibrium and the vehicle is resting on a flat level surface.

Aerodynamic reference point — a point in the *vehicle reference frame* that lies in the intersection of the *vehicle plane of symmetry* and the ground plane, mid-way between the front and rear axles, when the suspensions are in equilibrium and the vehicle is resting on a flat level surface.

Vertical position Z — $\mathbf{Z}_E$ coordinate of the *C.G.*

X position X — $\mathbf{X}_E$ coordinate of the *C.G.*

Y position Y — $\mathbf{Y}_E$ coordinate of the *C.G.*

Translational Motion

Lateral acceleration A_y — **Y** component of *acceleration vector* of the *C.G.* (SAE, ISO)[4]

Lateral velocity V_y — **Y** component of *velocity vector* of the *C.G.* (SAE, ISO)[4]

Longitudinal acceleration A_x— **X** component of *acceleration vector* of the *C.G.* (ISO)[4,5]

Longitudinal velocity V_x — **X** component of *velocity vector* of the *C.G.* (ISO)[4,5]

Vertical acceleration A_z — **Z** component of *acceleration vector* of the *C.G.* (ISO)[4,6]

Vertical velocity V_z — **Z** component of *velocity vector* of the *C.G.* (ISO)[4,6]

Angles

Aerodynamic sideslip angle β_{aero} — angle from **X** to velocity vector of air relative to the *vehicle reference frame*. (SAE)

Euler angles (ψ, θ, φ) — sequence of consecutive rotations about $\mathbf{Z}_E$, $\mathbf{Y}$, and $\mathbf{X}_V$ axes to convert from the *earth-fixed axis system* to the *vehicle axis system*. Note that φ is not identical to *roll* (φ_V). (SAE, ISO) The relationship is: $\varphi = \sin^{-1}(\sin(\varphi_V)/\cos(\theta))$

Pitch θ — angle from $\mathbf{X}$ to $\mathbf{X}_V$, about $\mathbf{Y}$. (SAE, ISO) θ can be calculated using a vector dot product: $\theta = -\sin^{-1}(\mathbf{X}_V \bullet \mathbf{Z}_E)$

Roll φ_V — angle from $\mathbf{X}_E \times \mathbf{Y}_E$ plane to $\mathbf{Y}_V$, about $\mathbf{X}$. (SAE, ISO)[3] Roll can be calculated using a vector dot product: $\varphi_V = \sin^{-1}(\mathbf{Y}_V \bullet \mathbf{Z}_E)$. It can also be calculated from the Euler angles θ and φ: $\varphi_V = \sin^{-1}(\cos(\theta)\sin(\varphi))$.

Sideslip angle β — angle from the $\mathbf{X}$ to the projection of the C.G. velocity vector onto the $\mathbf{X} \times \mathbf{Y}$ plane, about $\mathbf{Z}$.

Sideslip can be calculated from the lateral velocity V_y and longitudinal velocity V_x.

$$\beta = \tan^{-1}\frac{V_y}{V_x} \text{ (SAE, ISO)}$$

Yaw ψ — angle from $\mathbf{X}_E$ to $\mathbf{X}$, about $\mathbf{Z}$ (SAE, ISO)

Angular Velocity and Acceleration

Pitch acceleration α_y — $\mathbf{Y}$ component of *angular acceleration vector* of *vehicle reference frame*.[7]

Pitch velocity ω_y — $\mathbf{Y}$ component of *angular velocity vector* of *vehicle reference frame*.[8]

Roll acceleration α_x — $\mathbf{X}$ component of *angular acceleration vector* of the *vehicle reference frame*.[7]

Roll velocity ω_x — $\mathbf{X}$ component of *angular velocity vector* of *vehicle reference frame*.[8]

Yaw acceleration α_z — $\mathbf{Z}$ component of *angular acceleration vector* of the *vehicle reference frame*. (ISO)

Yaw velocity ω_z — $\mathbf{Z}$ component of vehicle *angular velocity vector* of the *vehicle reference frame*. (SAE, ISO)

Aerodynamic Forces and Moments

Forces and moments acting from the air on the vehicle are summed into a single resultant aerodynamic force vector, and a single resultant aerodynamic moment vector taken about the *aerodynamic reference point* .

Aerodynamic lateral force F_{yaero} — $\mathbf{Y}$ component of aerodynamic resultant force. (SAE)

Aerodynamic longitudinal force F_{xaero} — $\mathbf{X}$ component of aerodynamic resultant force. (SAE)

Aerodynamic vertical force F_{zaero} — $\mathbf{Z}$ component of aerodynamic resultant force. (SAE)

Aerodynamic pitch moment M_{yaero} — **Y** component of aerodynamic resultant moment. (SAE[18])

Aerodynamic roll moment M_{xaero} — **X** component of aerodynamic resultant moment. (SAE[18])

Aerodynamic yaw moment M_{zaero} — **Z** component of aerodynamic resultant moment. (SAE[18])

Suspensions and steering

For solid axles, the term *suspension* normally refers to the suspension for both sides of the axle. For independent suspensions, the term *suspension* refers to one side. The term *axle suspension* always refers to both sides.

Size and Weight

Track L_{TK} — distance between the *centers of tire contact* for one axle. In case of dual wheels, the midpoints of the *centers of tire contact* for each side are used. Track typically varies slightly with suspension *jounce*. (ISO)

Unsprung weight — portion of weight supported by a tire that is considered to move with the wheel. This usually includes a portion of the weight of the suspension elements. (SAE)

Kinematics

Camber — outward angular lean of wheel relative to *vehicle reference frame*: angle from $\mathbf{Z}_V$ to the $\mathbf{X}_W \times \mathbf{Z}_W$ plane. Regardless of the choice of coordinate systems, outward lean is positive. The symmetric sign convention is convenient for describing certain kinematical and compliance relationships for both sides of the vehicle. (SAE, ISO)[9]

Compliance camber — portion of *camber* due to tire forces (except vertical) and moments. (SAE, ISO)

Compliance steer δ_c — portion of *steer* due to tire forces (except vertical) and moments. (SAE, ISO)

Damper mechanical advantage R_d — Ratio of damper compression per unit of wheel jounce. This ratio is usually less than unity.

Driver steer δ_d — portion of *steer* due to steering wheel angle, with no forces or moments applied by the ground to the tires, and with no suspension movement.

Jounce — vertical movement of wheel or axle relative to the *vehicle reference frame*. Jounce is positive for compressive movement (wheel moving up relative to the body). There is no standard definition of zero jounce. (SAE[17])

Kinematical camber — *camber* measured with no tire forces or moments other than vertical. (Also defined as *camber* minus *compliance camber*.)[9,10]

Kinematical steer δ_k — *steer* measured with zero steering wheel angle and no tire forces or moments other than vertical. (Also defined as steer *angle* minus *compliance steer* and minus *driver steer*.)[9,10]

Pitch center — imaginary point in the $\mathbf{X}_V \times \mathbf{Z}_V$ plane through the lateral center of the *vehicle reference frame* at which a longitudinal force applied to the vehicle body is reacted without producing suspension jounce (front or rear). An alternate definition is that the pitch center is the intersection of the two lines shown in Figure B.4. Note: this definition of pitch center does not take into account the "wind up" effects of drive train torque applied to the wheels from the vehicle body.

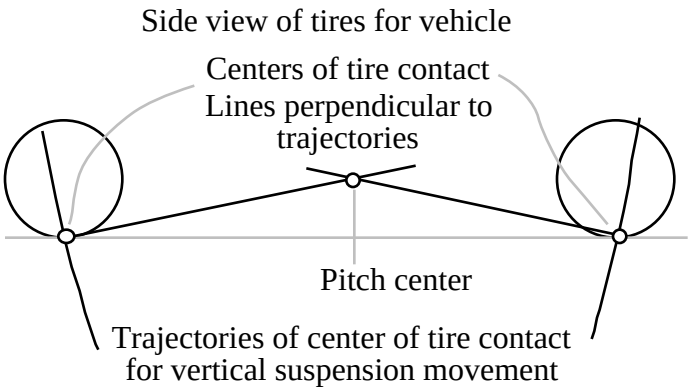

Side view of tires for vehicle
Centers of tire contact
Lines perpendicular to trajectories
Pitch center
Trajectories of center of tire contact for vertical suspension movement

Figure B.4. Pitch center.

Roll center — imaginary point in the $\mathbf{Y}_V \times \mathbf{Z}_V$ plane containing the two *wheel centers* of an axle, at which a lateral force applied to the vehicle body is reacted without producing a *suspension roll angle*. (ISO, SAE) An alternate definition is that the roll center is the intersection of the two lines shown in Figure B.5. (Note: the figure shows a non-equilibrium position of the vehicle.)

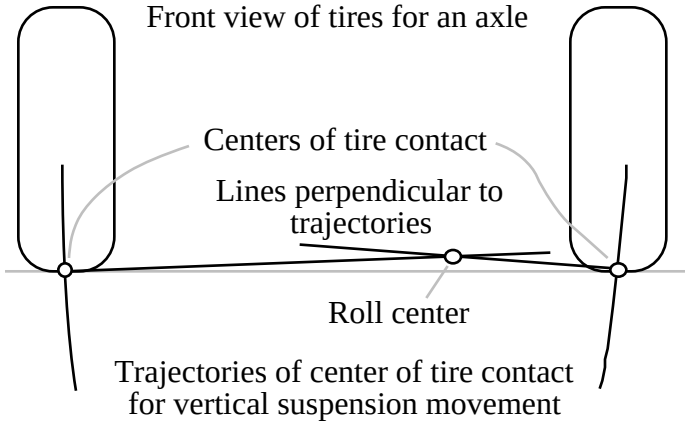

Front view of tires for an axle
Centers of tire contact
Lines perpendicular to trajectories
Roll center
Trajectories of center of tire contact for vertical suspension movement

Figure B.5. Roll center.

Spring mechanical advantage R_S — ratio of spring compression per unit of wheel jounce. This ratio is usually less than unity.

Steer δ — angle from $\mathbf{X}$ to $\mathbf{X}_W$, about $\mathbf{Z}$. (SAE, ISO)

Suspension roll angle — angle from line joining the *wheel centers* of an axle to the $\mathbf{X}_V \times \mathbf{Y}_V$ plane in the *vehicle reference frame*. (SAE, ISO)

Toe — inward steer of wheel relative to the *vehicle reference frame*: angle from $\mathbf{X}$ to $\mathbf{X}_W$. Regardless of the choice of coordinate systems, inward steer is positive. The symmetric sign convention is convenient for describing certain kinematical and compliance relationships for both sides of the vehicle. (SAE, ISO)[11]

Forces and Moments

Auxiliary roll moment M_{aux} — the *suspension roll moment* minus the moments due to the *suspension forces* from the two sides. A positive moment causes positive vehicle roll.

Damping force F_d — compressive force applied to the vehicle body by a damper.

Spring force F_s — compressive force applied to the vehicle body by a suspension spring.

Suspension roll moment M_{roll} — total static roll moment applied to *sprung mass* due to *suspension roll angle*. A positive moment causes positive vehicle roll.

Tires and wheels

Kinematics

Center of tire contact — point at intersection of a line passing through the wheel center and the ground, where the line is parallel with $\mathbf{Z}_W$. (The *center of tire contact* is not necessarily at the center of the tire contact patch. See Figure B.3.) (SAE, ISO)

Inclination γ — lean (angle) of wheel relative to *ground plane*: angle from $\mathbf{Z}_G$ to $\mathbf{Z}_W$, about $\mathbf{X}_G$. (SAE, ISO[12])

Longitudinal slip — the ratio: $\dfrac{\omega - \omega_0}{\omega_0}$

where ω is the *angular velocity* of the wheel about its *spin axis* and ω_0 is the free rolling angular velocity of the wheel that would be measured at zero *slip angle* and zero *inclination*. (ω_0 is the longitudinal velocity of the wheel center, divided by the effective circumference of the tire at that speed and load condition.) (SAE, ISO)[13]

Slip angle α — angle from $\mathbf{X}_G$ to the *velocity vector* of the *center of tire contact*, about $\mathbf{Z}_G$. (See Figure B.3.) (SAE, ISO)

Spin axis $\mathbf{Y}_W$— axis of rotation of wheel about spindle. (See Figure B.3.) (SAE)

Wheel center — intersection of *spin axis* and *wheel plane*. (See Figure B.3.) (SAE, ISO)

Wheel plane — central plane of wheel, normal to the *spin axis*. (See Figure B.3.) (SAE, ISO)

Forces and Moments

Forces and moments acting from the ground on the tire are summed into a single resultant force vector and a single resultant moment vector taken about the *center of tire contact*.

Aligning moment M_z — $\mathbf{Z}_G$ component of ground resultant moment. (SAE, ISO)

Overturning moment M_x — $\mathbf{X}_G$ component of ground resultant moment. (SAE, ISO)

Rolling moment M_y — $\mathbf{Y}_G$ component of ground resultant moment. (SAE[14], ISO)

Driving moment— $\mathbf{Y}_W$ component of moment applied by the vehicle to the wheel about the spin axis. (SAE[14])

Lateral tire force F_y — $\mathbf{Y}_G$ component of ground resultant force. (SAE, ISO[15])

Longitudinal tire force F_x — $\mathbf{X}_G$ component of ground resultant force. (SAE, ISO[15])

Vertical tire force F_z — $\mathbf{Z}_G$ component of ground resultant force. (SAE[16], ISO[15])

Notes

1. The ISO “wheel” axis system is similar to the “ground” system defined in this document, except ISO does not account for inclined ground surfaces.
2. Neither SAE nor ISO is explicit about the reference frame for the vehicle coordinate system, although both clearly involve the body.
3. SAE and ISO call this angle “vehicle roll,” and both use the term roll to denote and Euler angle. (This appendix emphasizes vehicle roll, which is measurable, whereas the Euler angle is not.)
4. Neither SAE nor ISO define the point whose velocity and acceleration vector are being described.
5. SAE defines “longitudinal” velocity and acceleration as the $\mathbf{X}_V$ component, and “forward” velocity as the **X** component. The **X** component of acceleration is not covered.
6. SAE defines “normal” velocity and acceleration using the $\mathbf{Z}_V$ axis. The **Z** components of acceleration and velocity are not covered.
7. SAE does not define angular accelerations; ISO defines them as second derivatives of Euler angles.
8. SAE defines angular velocities about the $\mathbf{X}_V$, $\mathbf{Y}_V$ and $\mathbf{Z}_V$ axes; ISO defines them as derivatives of Euler angles (only the roll derivative is measurable).
9. SAE and ISO define camber relative to $\mathbf{Z}_E$, rather than $\mathbf{Z}_V$.
10. ISO defines “camber angle change due to wheel travel kinematics” and “steer angle change due to wheel travel kinematics.”

11. ISO and SAE define both a toe angle and a toe displacement. SAE does not define a sign convention for toe angle.
12. The ISO definition of inclination is relative to absolute vertical ($\mathbf{Z}_E$) rather than the vector normal to the ground ($\mathbf{Z}_G$).
13. SAE defines longitudinal slip with units of percentage (a factor of 100 higher than the ISO definition). ISO uses the symbol S_{Xw} instead of K (K is used by Pacejka).
14. SAE uses the names “aligning torque,” “rolling resistance moment,” and “wheel torque” for “aligning moment,” “rolling moment,” and “driving moment,” respectively.
15. ISO uses the names “lateral force at wheel,” “longitudinal force at wheel,” and “vertical force at wheel.”
16. SAE uses the name “normal force” for negative force (relative to the SAE **Z** direction) and “vertical load” for the negative of “normal force.”
17. SAE calls compressive suspension deflection “compression.”
18. SAE does not define the point about which aerodynamic moments are taken.

Appendix C — ERD File Format

The ERD file format was developed by the Engineering Research Division (ERD) of the University of Michigan Transportation Research Institute (UMTRI) to facilitate automated plotting of simulation data, experimentally measured data, and data from various analysis programs.

A freely available plotter called EP (Engineering Plotter) has been developed for viewing data in ERD files. Versions of EP exist for the Mac and Windows platforms. The Windows version is called WinEP.

The animator program embedded in CarSimEd is also designed to work with ERD files.

An ERD file contains two independent sections, the header and the data. The header contains only text, and the data section contains only numbers. The numbers can be written in either text or binary form. The text form is convenient for viewing and editing data with a word processor, whereas the binary form provides more efficient access for automatic processing. If the data section is in text format, then both the header and the data are kept in a single file. However, if the data are in binary format, two files are used. The header is in an ordinary text file with the extension ERD, and the data are in a file with the name of the header file and the extension BIN. For example, if the header file is named `Out.erd` the name `Out.bin` must be used for the data file. For use in the plotter and animator programs in CarSimEd, both files must lie in the same folder.

The Header

The ERD file header consists of a series of conventional text lines that are human readable. These lines contain the information used by post-processing tools to read the numerical data.

Required Lines

As a minimum, the header contains three lines of text. The first line identifies the file as following the ERD format. The second line describes the way that the numerical data are stored in the data section of the file. The third required line is an END statement that indicates the end of the header portion. Any number of optional lines can be included between line #2 and the END line. Table C.1 summarizes the lines in an ERD file, and describes the parameters used in line #2 to describe the numerical data.

The second line of the file shown in Listing C.1 shows that the file contains data for 2 channels, with 529 samples/channel, stored as 1 binary record consisting of 4232 bytes, that the data storage format is type 1 (4-byte binary), that the interval between samples is 1.00, and that the status of the auxiliary numbers is -1.

Table C.1. Summary of records in an ERD file header.

Line No.	Description
1	`ERDFILEV2.00` — identifies file as having ERD format
2	NCHAN, NSAMP, NRECS, NBYTES, KEYNUM, STEP, KEYOPT — use commas to separate numbers NCHAN [integer] = Number of data channels NSAMP [integer] = Number of samples for each channel. The total number of sampled values in the data portion of the file is NCHAN × NSAMP. (If unknown, use –1.) NRECS [integer] = Number of records of data. (record ≈ line) Ignored for text data (KEYNUM = 5.) (If unknown, use –1.) NBYTES [integer] *binary data:* Number of bytes per record. If KEYNUM=0,1, or 5, this should be chosen such that each record begins with channel 1: that is, NBYTES = K × NCHAN × B, where K is an integer and B is the number of bytes/number (B=2 for integer, B=4 for floating-point). If KEYNUM=10,11,or 15, this should be NSAMP × B. *text data:* Number of samples per record. Thus each record contains NBYTES × NCHAN numbers (for KEYNUM=5). KEYNUM[integer] Indicates how the data are stored. 0, 10 = 2-byte integer (binary), 1, 11 = 4-byte floating point (binary), 5, 15 = Formatted floating-point (text). The format must be specified using the FORMAT keyword. For KEYNUM=0,1, and 5, the data are stored with all channels for the first sample together, then all channels for the second sample, etc. For KEYNUM=10,11, and 15, the data are stored with all samples for the first channel together, then all samples for the second channel, etc. STEP [real] = sample interval (e.g., time step) KEYOPT [integer] = auxiliary number used by some programs
•	**Optional records**. Each record begins with an 8-character keyword, followed by information associated with that keyword. Table C.2 lists keywords that have been used to date.
Last Line	`END` — indicates the end of the header

Listing C.1. Short Header for an ERD File with Binary Data.

```
ERDFILEV2.00
2, 529, 1,  4232,  1,  1.00000,  -1,
TITLE   1993 RPUG Study, Dipstick, Section 1, Measurement 1
SHORTNAMLElev.  RElev.
UNITSNAMft      ft
XLABEL  Distance
XUNITS  ft
END
```

Listing C.2 shows a longer header for a file with its data in text form. Note that the data begin immediately after the END line of the header.

Listing C.2. Typical Header for an ERD File with Text Data.

```
ERDFILEV2.00
2, 529, 1,  4232,  1,  1.00000,  -1,
TITLE   1993 RPUG Study, Dipstick, Section 1, Measurement 1
SHORTNAMLElev.  RElev.
LONGNAMELeft Elevation                  Right Elevation
UNITSNAMft      ft
GENNAME Profile Elevation               Profile Elevation
XLABEL  Distance
XUNITS  ft
FORMAT  (2G14.6)
PROFINSTDipstick
HISTORY Converted to ERD format at 23:46, Oct. 23, 1994
END
0.000000      0.000000
 0.416667E-03 -0.141667E-02
0.416667E-03  0.583333E-03
0.666667E-03  0.916667E-03
0.133333E-02  0.133333E-02
0.750000E-03 -0.166667E-02
-0.300000E-02 -0.458333E-02
-0.558333E-02 -0.500000E-02
-0.625000E-02 -0.658333E-02
-0.775000E-02 -0.825000E-02
```

Optional Lines with Keywords

Optional lines in the header begin with an eight-character keyword that defines a particular type of data contained in the remainder of the line. Keywords are associated with one of five general data types: integers, floating point (real) numbers, 8-character names, 32-character names, and 80-character names. The number of data items is either

one per file (e.g., TITLE of data set in the file), one per channel (e.g., a short name for each channel), an arbitrary number N (e.g., static axle loads for N axles), or repeatable.

Table C.2 lists common keywords recognized by most post-processing tools. The use of some of these keywords is demonstrated in Listing C.1 and Listing C.2.

Table C.2. Keywords in ERD file.

keyword	**Description**	**No. of Values**	**Variable Type**
VERSION	Line 1 in header file.	1	char*32
<*none*>	Line 2 in header file. See Table C.2 for details.	7	int, real
&n	Continuation keyword, indicates that the previous line ended in column *n* and is continued in this line in column 9. Used to break long lines into multiple short lines.		
(The following 6 lines are used by EP and are recommended for inclusion in all ERD files)			
GENNAME	Generic names for variables, used for labeling Y axis when several variables are plotted on the same axis (e.g., Force).	NCHAN	char*32
LONGNAME	Long names for channels.	NCHAN	char*32
SHORTNAM	Short names for channels.	NCHAN	char*8
TITLE	Title used for file.	1	char*80
UNITSNAM	Units of channels.	NCHAN	char*8
XUNITS	Units of independent variable (e.g., sec).	1	char*8
(The following line is required for EP to create Channel 0*, e.g., time)*			
XLABEL	Name of ind. variable in ERD file (e.g., time).	1	char*32
FORMAT	FORMAT statement for text data. Ex: (4F10.4)	1	char*32
GAIN	Gains for channels. (Default = 1.) Usually required for integer*2 data.	NCHAN	real
OFFSET	Offsets for channels. (default = 0.) Usually required for integer*2 data.	NCHAN	real
PROFINST	Instrument or model associated with data	1	char*32
RIGIBODY	Body or part associated with each channel	NCHAN	char*32
SPEEDMPH	Speed associated with data, in mile/hr.	1	real
TESTID	Number used to identify a test.	1	real
XSTART	Starting value of ind. variable. At each sample i, the X value is: X = (i-1) * STEP + XSTART	1	real

Often, names associated with a keyword are shorter than the space allowed. When more than one name is on the same line, the names are padded with blanks as needed so that

following names begin at the correct column positions. For example, the header shown in Listing C.1 includes names of the units for each channel, as identified with the keyword UNITSNAM. The name of units for the first channel, ft, has only two characters. Thus, it is followed by six spaces so that the name for the second channel, ft, begins in the correct column position.

The Data Section

The data section of the ERD file contains nothing but numbers, organized into columns and rows. The form in which the numbers are stored depends on the value of the KEYNUM parameter from line 2 of the header (see Table C.1). The total number of values that will appear in the data section is NCHAN × NSAMP. All of the numbers in the data portion are stored in the same format, and there can be no missing values.

Text Data

The text format can be used for transporting data in ERD files between different computers, and sometimes even for reading the same file with different programs on the same computer. It is also convenient when numbers are typed in manually, or when numbers are to be edited using a text editor. However, there are penalties for using text representations of numbers. First, the computer must work hard to translate the text numbers into binary form. It takes about 10 times longer to read a text file than a binary equivalent. Second, text files take up much more disk storage than binary files.

When data are stored in text form, the numbers are kept in the same file as the header, with the numbers beginning immediately after the header. The ERD file in Listing C.2 shows an example of numerical data in text form.

Another option is available when the numbers are always separated by delimiters such as spaces or commas. This occurs when the numbers are obtained by a commercial analysis program or when they are "captured" from another computer. The file of numbers can be made into an ERD file by inserting a 3-line header at the beginning of the file.

If the header of the ERD file does not contain a line with the FORMAT keyword, it is the same as if the FORMAT is a blank. When this occurs for a text file, the file is assumed to contain numbers in free form. The only restriction on free format numbers is that adjacent numbers must be separated. For example, the following line is valid for representing 5 numbers: `1.2000 3 4 -2.01E-01 14.3`

The following line is not, because the third and fourth numbers touch.

```
1.2000 3.0000 4.0000-2.01E-01 14.3000
```

Numbers may be separated by one or more spaces, the tab character, or a comma.

Binary Data

Reading and writing binary data is efficient because the computer does not need to perform any conversions or transformations as the data values are moved between the file

and the computer memory. When a binary format is used, the data portion of an ERD file is a direct copy of a portion of the computer memory, corresponding to a two-dimensional array having dimensions sized to the number of channels and the number of samples. As indicated in Table C.1, two forms of binary data are currently supported: 2-byte integer and 4-byte floating point. 2-byte integer data are typically obtained by data-acquisition systems. Each integer value is a sampled reading obtained from a digitizer during a test. For most engineering applications, data are stored (in the computer memory) in 4-byte floating point format, also known as single-precision floating point. The 4-byte floating point format is commonly used for data generated by computer. The maximum efficiency for data processing is usually obtained when the 4-byte floating point format is used.

The ERD file format is used on a variety of computer systems and for a variety of mass storage media. On some systems, binary data are stored in discrete records. A computer program reading such a file needs to know how many bytes each record contains, and how many records are in the file. Thus, the header contains these two parameters.

Disk files on workstations and desktop computers are not structured: a binary file is simply a continuous stream of bytes that continues to the end of the file. Thus, technically correct parameter values for the header could be one record, containing all of the bytes for the file. Also, there is a certain amount of overhead associated with reading a record. The time needed to read the data for a file is minimized if a single read operation is performed for the entire file.

On the other hand, if the file is large, the memory needed to read the entire file in one chunk may not be available with some programs. A second problem can occur if the true number of bytes in the file is less than the number as inferred by the parameters NBYTES and NRECS (i.e., the total size of the file should be NBYTES x NRECS). The last "record" is not read, resulting in a loss of data. If the records are large, this loss could be significant. These problems are reduced if a value of NBYTES is specified such that it divides the data into NRECS records of smaller chunks of data.

Appendix D — Plotter Files and Keywords

When WinEP is run from CarSimEd, it receives the information that describes a plot or set of plots from a text file. The passing of information between a CarSimEd data screen and WinEP is automatic, and the details of the files used to transfer information do not have to be understood by the user for normal use of CarSimEd. The information in this appendix is provided to aid in debugging, and for those wishing to use WinEP outside of CarSimEd.

PLT Batch Control Files

Most plots made in CarSimEd are generated when you click the **Plot** button on the **Runs** screen or the **Plot Setup: Batch** screen. When this occurs, the CarSimEd library creates a text file with the extension PLT and sends that file to WinEP. The PLT file is similar to the PAR files used in CarSimEd to transfer information from the database to the simulation solver programs. It is a plain ASCII text file that contains keywords with file names, variables names, and other pieces of information that tell WinEP how to make a plot.

Listing D.1 shows an example `PLT` file and Table D.1 summarizes the format.

Listing D.1. Example PLT batch plot file.

```
FILELIST
FILENAME C:\CARSIMED.45\RUNS\894.ERD
FILENAME C:\CARSIMED.45\RUNS\808.erd
PARSFILE C:\CARSIMED.45\plot\setup\814.par
RUN Screen
FILENAME C:\CARSIMED.45\RUNS\894.ERD
FILENAME C:\CARSIMED.45\RUNS\808.erd
PARSFILE C:\CARSIMED.45\plot\setup\795.par
RUN Screen
FILENAME C:\CARSIMED.45\RUNS\894.ERD
FILENAME C:\CARSIMED.45\RUNS\808.erd
PARSFILE C:\CARSIMED.45\plot\setup\817.par
RUN Screen
FILENAME C:\CARSIMED.45\RUNS\894.ERD
FILENAME C:\CARSIMED.45\RUNS\808.erd
PARSFILE C:\CARSIMED.45\plot\setup\810.par
RUN Screen
END
```

The lines in the PLT file are grouped into plot sets, separated by lines beginning with the RUN keyword. For example, Listing D.1 has four sets, defining four plots. Lines within a plot set can be arranged in any order.

The PLT file is normally read by WinEP when you click a **Plot** button from the **Runs** screen or from the **Plot Setup: Batch** screen. Although less common, you can also read one interactively in WinEP by selecting the **File** menu option **Load Batch File**.

Table D.1. PLT batch plot file format.

Keyword	Instruction to Program
FILELIST	Line 1: Identify this as a list of plot descriptions*.
FILENAME *ERDfile*	Read the file *ERDfile* for data to plot. *ERDfile* should follow the ERD format. (Simple text files with tables of numbers can also be specified.)
PARSFILE *filename*	Open the file *filename* and read a parsfile with information about what variables to plot, format, transforms, etc.
RUN screen	End the plot set.
END	Stop reading data from the PLT file.

* In order to support real-time plotting in other software packages similar to CarSimEd, this keyword is no longer tested by WinEP. Anything on line 1 will be accepted.

Plot Setting Files

A plot setting file contains a list of variables to extract from the data files, along with formatting information. Setting files are normally created automatically by the CarSimEd library **Plot Setup: Single** and are stored in the same PARSFILE format used throughout CarSimEd. Listing D.2 shows an example PARSFILE with information to define a plot.

Listing D.2. Example plot template file.

```
PARSFILE
PLOTCHANNELS Vx,Time,
PLOTCHANNELS Vx_LF,Time,
PLOTCHANNELS Vx_LR,Time,
PLOTCHANNELS Vx_RF,Time,
PLOTCHANNELS Vx_RR,Time,
PARSFILE C:\CARSIMED.45\PLOT\FORMAT\31.par

PTITLE Vx -- wheel speeds
END
```

Note: Because a PARSFILE can reference other PARSFILEs, the list of channels and the plot formatting can be spread over several

files. For example, formatting information can be placed in the template file or in the format file described below.

Plot Transform Files

WinEP supports modest transformations of the data being plotted. Offsets can be added to each variable (X and Y). The variables plotted on the Y axis can also be filtered with a moving average to smooth the data (low-pass filtering), remove low-frequency content (high-pass filtering), both (band-pass filtering), or neither.

WinEP can plot up to 20 data sets, meaning that there are potentially 40 offsets to specify. Listing D.3 shows an example PARSFILE with a few of the 40 offset keywords. If not specified, the default offset is 0.

Listing D.3. Plot transform settings.

```
PARSFILE
FILTER BANDPASS 10, 1

OFFSET_X1 0
OFFSET_Y1 0
OFFSET_X2 0
OFFSET_Y2 0
OFFSET_X3 0
…
PTITLE Vx -- wheel speeds
END
```

The valid values associated with the keyword `FILTER` are: `None`, `Hipass`, `Lowpass`, and `Bandpass`. For a value other than `None`, the baselength(s) associated with the filter should follow on the same line.

Note: Because a PARSFILE can reference other PARSFILEs, the list of channel offsets can be spread over several files.

Plot Format Files

The initial appearance of a plot is read from a format file. Format files are plain ASCII text files which contain keywords with parameters that tell WinEP how to format a plot. All format files follow the PARSFILE format and have the extension PAR. The keywords used in a format file are described in Table D.2.

Table D.2. Format keywords and descriptions.

Keyword	**Description**
PARSFILE	Specifies the start of the format file.
END	Specifies the end of the format file.
XLINEAR *type*	X axis type. Valid values are `log` and `linear`.
YLINEAR *type*	Y axis type. Valid values are `log` and `linear`.
XMAXMIN *type*	Either a keyword specifying that the X axis is auto-scaled, or two numbers: the minimum and maximum values for the axis.
YMAXMIN *type*	Either a keyword specifying that the Y axis is auto-scaled, or two numbers: the minimum and maximum values for the axis.
SYMBOLS *s1,s2, ...,s20*	Symbols used for each X-Y data set. There must be a total of 20 symbols indicated. Symbol key: 0 = No symbol 1 = Square 2 = Triangle 3 = Diamond 4 = Cross X 5 = Plus 6 = Circle
LINESTYL *s1,s2,s3,...,s20*	Line style for each X-Y data set. There must be a total of 20 styles indicated. Line style key: 0 = No Line 1 = Solid Line 2 = Dotted Line 3 = Heavy Solid Line 4 = Dash Line 5 = Dash Dot Line 6 = Dash Dot Dot Line
TITLEFONTSIZE *ps*	Size of the title font: ps = font point size.
TITLEFONTNAME *name*	Name of the title font (Arial, etc.).
TITLEFONTSTYLE *style*	Style of the title font (Regular, Italic, Bold, Bold Italic)
LEGENDEFONTSIZE *ps*	Size of the legend font: ps = font point size.
LEGENDFONTNAME *name*	Name of the legend font (Arial, etc.).
LEGENDFONTSTYLE *style*	Style of the legend font (Regular, Italic, Bold, Bold Italic)
LABELFONTSIZE *ps*	Size of the axis label font: ps = font point size.
LABELFONTNAME *name*	Name of the axis label font (Arial, etc.).
LABELFONTSTYLE *style*	Style of the axis label font (Regular, Italic, etc.)
TICLABELFONTSIZE *ps*	Size of the axis numbers: ps = font point size.
TICLABELFONTNAME *name*	Name of the axis number font (Arial, etc.).
TICLABELFONTSTYLE *style*	Style of the axis number font (Regular, Italic, etc.)

LEGENDLABEL *b1, b2, b3, b4, b5, b6*	Names to use in creating the legend. Values are 0 (do not use) or 1 (use). The sequence is: b1= shortname, b2 = longname, b3 = genname, b4 = rigidbody name, b5 = file title, b6 = file name.
LEGENDLOCATION *name*	Location in plot window for legend. Options are `AutoLocation`, RightOfPlot, `OnPlotUpperLeft`, `OnPlotUpperRight`, `OnPlotLowerLeft`, and `OnPlotLowerRight`
GRID *type*	Type of grid. Options are `NoGrid`, `CoarseGrid`, and `FineGrid`.
COLORS *c1,c2,c3,...,c20*	Color for each channel. There must be a total of 20 colors indicated. Color key: 0 = Black 1 = Maroon 2 = Red 3 = Green 4 = Bright Green 5 = Dark Blue 6 = Light Gray 7 = Light Blue 8 = Blue 9 = Muddy Yellow 10 = Yellow 11 = Gray 12 = Purple 13 = Teal 14 = Bright Purple

To apply the format settings from a file, load the format file using the **File** menu command **Load Plot Format**.

To create a format settings file, use the **File** menu command **Save Plot Format**.

Preference File Format

The preference file `Epprefs.txt` is used to set and save information that is retained between WinEP sessions. This information includes the window positions and paths to directories. The information can be viewed and edited using the **Edit** menu command **Preferences**. Listing D.4 shows an example preferences file.

Listing D.4. Example Preferences file.

```
ep-prefs
temp-folder-path C:\CARSIMED.45\PLOT\tmp
last-datafile-path C:\CARSIMED.45\PLOT
last-batchfile-path C:\CARSIMED.45\PLOT\tmp
last-formatfile-path C:\CARSIMED.45\PLOT\format
last-imagefile-path C:\CARSIMED.45\PLOT\image
startup-window-mode tile-vert
startup-toolbar-mode false
plot-window-position 10,10,727,550
end
```

The preferences file is created automatically by WinEP whenever the program quits, so it is not necessary to be concerned with its format.

Text Files

Although WinEP is mainly intended to plot data contained in ERD files, it is also capable of reading numerical data from tables stored in plain text files. Plain text files can be made by the solver programs in CarSimEd to simplify the importing of data into spreadsheets or other programs. If this form of output is chosen, WinEP can still be used to view the results (althought the labeling is not as nice). Also, WinEP can be used to view data exported from spreadsheets or other mathematical packages.

Listing D.5 shows part of a text file that can be read by WinEP. The first line should contain labels, and following lines should have numbers. The numbers should be separated by at list one white space and/or commas.

Listing D.5. Portion of a text file with data to plot.

```
"Time",  "AAy_LF  ",  "AAy_LR  ",  "AAy_RF  ",  "AAy_RR  ", …
     0.000000,      0.000000,      0.000000,      0.000000, …
     0.050000,     -0.041291,     -0.040227,      0.041170, …
     0.100000,     -0.096683,     -0.096122,      0.096175, …
     0.150000,     -0.098988,     -0.104285,      0.098771, …
     0.200000,     -0.073881,     -0.071838,      0.073669, …
     0.250000,     -0.050576,     -0.050680,      0.049884, …
     0.300000,     -0.032560,     -0.032437,      0.031484, …
     0.350000,     -0.019510,     -0.019438,      0.018365, …
     0.400000,     -0.010514,     -0.010540,      0.009256, …
     0.450000,     -0.004606,     -0.004626,      0.003102, …
     0.500000,     -0.000949,     -0.000990,     -0.000582, …
     0.550000,      0.001096,      0.001043,     -0.002610, …
     0.600000,      0.002020,      0.001971,     -0.003566, …
     0.650000,      0.002229,      0.002178,     -0.003693, …
```

Appendix E — Animator Files and Keywords

The animator program reads two kinds of input files, as indicated in Figure E.1. All of the settings specific to the animator are read from keyword-based text files, typically with the extension PAR. These files follow the PARSFILE format used throughout CarSimEd. A single top-level PARSFILE contains the names of other PARSFILEs with camera information, vehicle information, reference frames, etc.

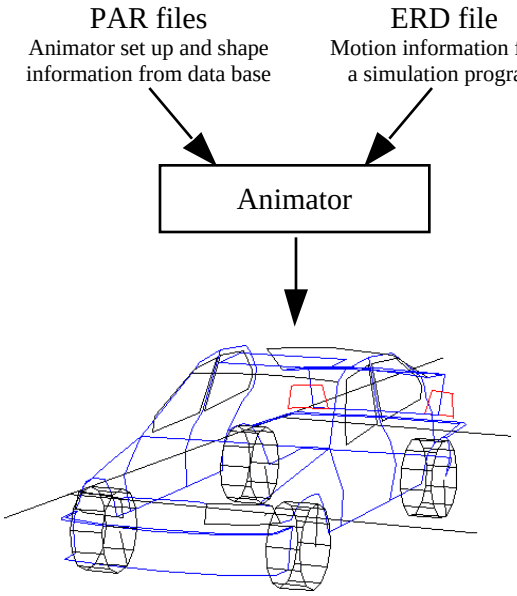

PAR files
Animator set up and shape information from data base
ERD file
Motion information from a simulation program
Animator

Figure E.1. Animator input files.

The motions predicted in a simulation are defined by variables that are written in the ERD file that is written by the solver programs whenever a run is made. Appendix C provides details of the ERD file format, and Appendix J shows a list of variables contained in a typical CarSimEd ERD file.

Overview of a PARSFILE

Most information about objects and how they are viewed is read from an input PARSFILE. Listing E.1 shows part of a PARSFILE.

Listing E.1. PAR file with some animators settings.

```
parsfile
; ** parsfile generated: 10/13/99 12:04:09 **

set_camera_reference_frame camera tracking x-y-z
set_camera_x -1.000000
set_camera_y 50.000000
set_camera_z 0.000000

set_focal_length 0.100000

set_lookpoint_reference_frame body
set_lookpoint_x -1.500000
set_lookpoint_y 0.000000
set_lookpoint_z 0.000000

set_use_cpu_clock off
set_superimpose off
...
end
```

A PARSFILE can contain several kinds of information: (1) numerical constants such as coordinates of points, (2) names of variables in the ERD file that must be read by the animator, (3) names associated with objects (e.g., `body`), and (4) pathnames for linked PARSFILEs with more data. In all cases, white space immediately following the keyword is skipped, and the remainder of the line is interpreted as the value to be assigned to an internal variable. Spaces are significant after the first non-blank character following the keyword.

The PARSFILE is similar to those used by the solver programs. It should begin with a line containing the single keyword, `parsfile`. It should end with a line containing the single word `end`. However, the `end` keyword is optional. If not present, the animator reads to the end of the file. Blank lines and lines with unknown keywords are skipped when the file is read. Multiple lines with the same keyword are permitted.

The PARSFILE Keyword

Links to other PARSFILEs are provided with the `parsfile` keyword:

`parsfile` *pathname*

where *pathname* is the path to another PARSFILE. Through the use of these `parsfile` links, data for each object can be contained in a separate file. For example, each `part` data set can be placed in a separate file. The data set for a `reference-frame` can be in a file that contains a `parsfile` link to each part associated with the frame. The entire animation data can be just a set of `parsfile` links to the non-moving parts, followed by the data sets for the moving reference frames.

Organization of Animator Inputs

At the top level, the animation shape data should be organized by reference frame. When a shape is introduced, it must be associated with a specific reference frame. After a new reference frame is introduced, all following inputs are said to fall within the *scope* of that reference frame. For example, a shape is in the scope of a reference frame if its coordinates are defined by the animator as being fixed in that reference frame.

Overall, the animation input data should be organized as follows:

parsfile
<information about grid, camera, and target path>

< information associated with fixed frame>

add_reference_frame {*name1*}
< information about first moving reference frame>

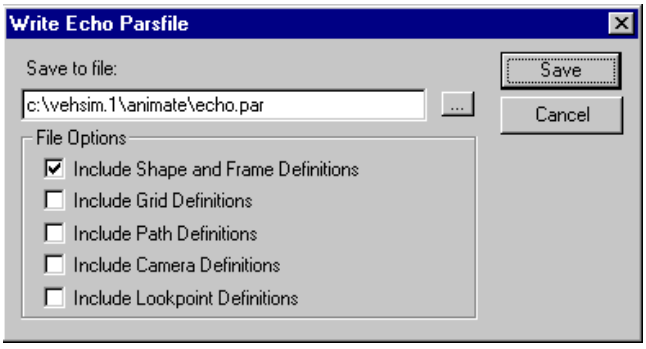
add_reference_frame {*name2*}
< information about second moving reference frame>

< more reference frame and associated information, as needed>

end

Making an Example PARSFILE

The animator has the capability of writing a single PARSFILE with all of the current settings, reference frames, and shapes. This is helpful for debugging and documenting an animator setup. To make a file, go to the **File** menu and select the item **Save Parsfile As...** This brings up a dialog box with check boxes associated with different kinds of information. Check all of the boxes and save the file.



Units, Coordinates, and Sign Conventions

The animator uses the sign conventions and Euler angle definitions described in Appendix B. (X is forward, Z is up, Y is to the vehicle left.)

All coordinates have units of length, and must be the same. In CarSimEd, all coordinates in the output file have the units of meters. Both variable and static coordinates can be converted using scale factors, if necessary. The camera focal length must also have the same units (meters).

All Euler angles must be expressed in degrees. Scale factors are used to convert to degrees if the variables in the ERD file have different units such as radians or revolutions.

Camera Settings

Listing E.1 on page 272 shows part of a PARSFILE that has the animator settings that are related most directly to the camera setup. Most of these are sent from the **Animator: Camera Setup** screen described in Chapter 9. Table E.1 lists the keywords that are used to specify camera settings.

Table E.1. Keywords for the animator camera settings.

Keyword	**Value**	**Description**
set_camera_reference_frame	name of reference frame	reference frame in which the camera is situated
set_camera_x set_camera_y set_camera_z	numbers	coordinates of the camera location in its reference frame
set_lookpoint_reference_frame	name of reference frame	reference frame in which the look point is situated
set_lookpoint _x set_lookpoint _y set_lookpoint _z	numbers	coordinates of the lookpoint in its reference frame
set_focal_length	number	focal length of camera (distance from point of viewer to 2D image on screen)
set_use_cpu_clock	`on` or `off`	option to slow animation down to real time by using the clock
set_superimpose	`on` or `off`	option to superimpose all images—don't erase between animation frames

Reference Frames

Understanding the concept of a reference frame is important if you wish to effectively use the animator. Chapter 6 introduces the concept and the Chapter 9 section **Animator: Reference Frames** explains how you define a reference frame within CarSimEd. (Appendix B also defines reference frame to the extent needed to define vehicle dynamics terminology.) Table E.2 lists keywords associated with reference frames.

Table E.2. Keywords for the animator camera settings.

Keyword	Value	Description
add_reference_frame	name of new reference frame	gives name to new reference frame and starts its scope
set_x_name set_y_name set_z_name	names of variables in ERD file	specifies the variables to be read from the ERD file and associated with X, Y, Z coordinates of the reference frame
set_pitch_name set_roll_name set_yaw_name	names of variables in ERD file	specifies the variables to be read from the ERD file and associated with Euler angles
set_scale_var_x set_scale_var_y set_scale_var_z set_scale_var_roll set_scale_var_pitch set_scale_var_yaw	numbers	scale factors for data read from the ERD file (default = 1.0)
set_offset_var_x set_offset_var_y set_offset_var_z set_offset_var_roll set_offset_var_pitch set_offset_var_yaw	numbers	offsets added to coordinates and Euler angles (default = 0.0)
set_euler_angles	`yaw_pitch_roll` or `yaw_roll_pitch`	sequence of rotation by Euler angles used to define orientation of the reference frame
x_ref_length y_ref_length z_ref_length	numbers	reference lengths associated with the coordinate of points in parts associated with the reference frame
x_length y_length z_length	numbers	actual lengths to be used by the animator to scale coordinates of points in parts within the reference frame

The keyword `add_reference_frame` has three effects:

1. It starts the scope of a new reference frame.
2. It ends the scope of the previous one.

3. It assigns a name to the new frame that can be used with the `set_camera_reference_frame` and `set_lookpoint_reference_frame` keywords (see Table E.1). Each reference frame must have a unique name.

Position and Orientation of the Reference Frame

Overall, a reference frame is defined by six variables: three coordinates (X, Y, and Z), and three Euler angles. The animator reads the required six variables from the output files generated by the solver programs. The six keywords used to specify ERD file short names determine how the three coordinates and three Euler angles are defined. White space between the keyword and the short name is ignored by the animator, and then the next eight characters are taken literally, including spaces. However, if the line ends before eight characters are read, the animator pads the short name with spaces until it is eight characters long. All six are optional. If not specified, the animator uses a value of zero for that variable.

After reading the six variables, each coordinate and Euler angle is calculated with a relationship of the form:

$$\text{coordinate} = C_o + C*SF_c$$

$$\text{angle} = A_o + A*SF_a$$

where C and A are the translation and angle variables obtained from the ERD file, C_o and A_o are the constant offsets, and SF_a and SF_c are scale factors (gains). The offsets and scale factors are specified by the keywords shown in Table E.2 (`set_offset_var_x`, `set_offset_var_y`, etc.) The scale factors are commonly used to convert angles to degrees (from radians or revolutions). The offsets are used in some other applications to convert relative coordinates to absolute coordinates.

The keyword `set_euler_angles` is used to specify the type of transformation used. There are two options: `yaw_pitch_roll` (used for vehicle sprung-mass reference frames) and `yaw_roll_pitch` (used for rolling-wheel reference frames).

Scope of the Reference Frame

The scope of a reference frame begins when the keyword `add_reference_frame` is encountered, and continues until this keyword is used again to start the scope of a different reference frame. All of the keywords shown in Table E.2 can be repeated several times in a file. Each time, the value associated with the keyword is applied only to the current reference frame, as defined by the previous `add_reference_frame` command.

Within the scope of each moving frame, you should specify one or more objects. The keywords `add_wheel` and `add_part` have the effect of starting the scope of a new object. They also have the effect of ending the scope of the previous object. However, they do not affect the scope of the current moving reference frame. In the example listing, descriptions of parts and wheels are contained in other PARSFILEs, identified with pathnames following the `parsfile` keyword.

Resizable Objects

The animator has a provision for re-scaling all parts within the scope of a reference frame. This is used in CarSimEd to stretch or shrink vehicle bodies based on wheelbase and track width. Separate scale factors are used for X, Y, and Z coordinates of all parts associated with the reference frame. The relation, shown for the X direction, is defined by a reference length and an actual length. For the X direction the keywords are `x_ref_length` and `x_length`. The conversion is:

$$X_{new} = X \bullet \frac{x\_length}{x\_ref\ \_length}$$

Where X_{new} is the newly scaled coordinate, X is the coordinate from the part or wheel description, x_length is the wheelbase used in the model and x_ref_length is the reference wheel base.

The scaling is done only if both the length and reference lengths are specified. Otherwise, the re-sizing is omitted and the original coordinates are used to define the points.

Parts (Shapes)

A part is a set of points connected by straight lines. Each point is defined by a set of three coordinates (X-Y-Z). The animator starts with the first point, and draws connecting lines to each following point in a list. Parts are called shapes in the graphical database, and are described in the Chapter 9 section **Animator: Shapes**.

All coordinates are assumed to be in a local coordinate system associated with the active reference frame.

There are four properties that can be set for a `part`: a name, a list of XYZ coordinates, line thickness, and color. Listing E.2 shows an example PARSFILE containing a complete `part` description. The keywords are defined in Table E.3.

Listing E.2. Part data description

```
parsfile
add_part hood
set_Color blue

set_coordinates
-0.6273 -0.7344 0.8721
-0.5355 -0.4131 0.8874
-0.4896 -0.0000 0.9027
-0.5355 0.4131 0.8874
-0.6273 0.7344 0.8721
0.0000 0.7191 0.7803
0.5202 0.6579 0.6732
0.6426 0.2601 0.6885
0.6579 -0.0000 0.6885
0.6579 -0.2601 0.6885
0.4590 -0.6885 0.7038
0.0000 -0.7191 0.7803
-0.6273 -0.7344 0.8721
end_coordinates

set_scale_x 1
set_scale_y 1
set_scale_z 1

set_offset_x 0
set_offset_y 0
set_offset_z 0

end
```

Table E.3. Keywords for describing parts.

Keyword	**Value**	**Description**
add_part	name of part	starts scope for new part (the name itself is treated like a comment)
set_color	color (9 choices): black, white, red, blue, green, magenta, yellow, dark gray, light gray	color used for lines drawn to connect the points in this part
set_line_width	integer	sets thickness of lines drawn for this part
set_coordinates end_coordinates	list of coordinates: 3 numbers per line	coordinates of the points making up the shape
set_scale_x set_scale_y set_scale_z	numbers	scale factors applied to all coordinates in the part
set_offset_x set_offset_y set_offset_z	numbers	offsets added to all points in the part

It is sometimes convenient to use the same part in more than one reference frame. Therefore, the name is not necessarily unique. The part name is used only to make the PARSFILE more readable, and possibly to provide more detailed error messages in future versions.

The color and line thickness default to values of black and 1, respectively. Therefore, the commands `set-line-width` and `set_color` are optional. There is no default set of coordinates, so the list must be provided. The list of coordinates begins with a line containing the keyword `set_coordinates`. Each following line should contain an X, Y, and Z coordinate, separated by white space, until the list ends with a line containing the keyword `end_coordinates`. Within the block defined by the keywords `set_coordinates` and `end_coordinates`, there must not be blank lines, comments, or other keywords. The coordinates are assumed to apply to the current reference frame.

The listed coordinates for the part are transformed by the equations:

$$x_{new} = x_o + s_x\, x$$

$$y_{new} = y_o + s_y\, y$$

$$z_{new} = z_o + s_z\, z$$

where x_o, y_o, and z_o are offsets and s_x, s_y, and s_z are scale factors specified with the keywords `set_offset_x`, `set_offset_y`, `set_offset_z`, `set_scale_x`, `set_scale_y`, and `set_scale_z`. The keywords for scale factors and offsets are optional. For each part, the scale factors default to values of 1 and the offsets default to 0. By setting a scale factor to -1, a user can easily mirror a part. The offsets allow a user to change the location of a part within the reference frame without calculating new values for the coordinates.

The scope of a part begins when the keyword `add_part` is encountered, and continues until another part, wheel, or reference frame is added. Values associated with most of the keywords shown in Table E.3 are applied to the current part, as defined by the previous `add_part` command.

Wheels

A wheel is approximated by drawing an object consisting of two polygons whose corresponding vertices are connected by lines. An optional radial line is drawn to show the wheel rotation angle.

There are five properties that can be set for a wheel: name, radius, thickness, line-thickness, color, and the presence of a radial line. All properties except the name have a default value, and therefore, the commands shown below `add_wheel` are optional. It is likely that the same wheel definition will be used several times, and therefore, the name is not necessarily unique.

Table E.4. Keywords for describing a wheel.

Keyword	Value	Description
add_wheel	name of wheel	starts scope for new wheel (the name itself is treated like a comment)
set_color	color (9 choices): black, white, red, blue, green, magenta, yellow, dark gray, light gray	color used for lines drawn to connect the points in this part
set_num_points	integer	sets number of points in polygon used to approximate a circle
set_radius	number	radius of wheel
set_thickness	number	thickness of wheel
set_line_width	integer	sets thickness of lines drawn for this part
set_radial_line	`on` or `off`	if `on`, draw radial line to help show how fast wheel is spinning
set_scale_x set_scale_y set_scale_z	numbers	scale factors applied to all coordinates in the part
set_offset_x set_offset_y set_offset_z	numbers	offsets added to all points in the part

The scope of a wheel begins when the keyword `add_wheel` is encountered, and continues until another part, wheel, or reference frame is added. Values associated with most of the keywords shown in Table E.4 are applied to the current wheel, as defined by the previous `add_wheel` command.

Grid and 3D Ground Surface

The animator will draw a grid fixed in the global reference frame. If not specified, then there is no grid. The grid can be flat, or it can be based on 3D shape data used for the ground in CarSimEd. Table E.5 lists the keywords and

Listing E.3 shows an example PARSFILE.

Table E.5. Keywords for describing parts.

Keyword	**Value**	**Description**
add_grid	<none>	tells animator to draw a reference grid
set_interval_x set_interval_y	numbers	spacing used for drawing the grid lines
set_color	color (9 choices): black, white, red, blue, green, magenta, yellow, dark gray, light gray	color used for the grid lines
set_min_x set_max_x set_min_y set_max_y	numbers	size of the grid in the X and Y directions (if the min and max are set equal, the grid is sized automatically in that direction)
z_ground_carpet endtable	first line: number of columns next lines: ground data	3D ground information, as used by vehicle models

Listing E.3. PARSFILE with grid information.

```
parsfile
; ** parsfile generated: 10/13/99 12:04:09 **

add_grid
set_interval_x 5.000000
set_interval_y 5.000000
set_color light gray
set_min_x -5.000000
set_max_x 135.000000
set_min_y -5.000000
set_max_y 5.000000

z_ground_carpet 3
          0.0,           -10,            10,
            0,             0,             0,
            5,             0,             0,
           10,           0.1,           0.1,
           15,           0.1,           0.1,
           20,             0,             0,
           25,             0,             0,
endtable
```

The 3D ground information is read from the same file used by the vehicle simulation programs. It is described in the Chapter 9 section **Input: 3D Ground Surface Elevation**.

The 3D ground is defined over the range of all possible X and Y values (it is not limited to the range covered in the table). The table lookup function used in the vehicle programs

is also used in the animator to generate coordinates at the intersection of the X and Y grid lines.

The grid spacing is not always compatible with the 3D ground data. If the z_ground_carpet table has closely spaced features, they will not be seen in the grid. To view detailed ground features, it is best to create a part description associated with a non-moving reference frame.

The scope of the grid information is unlimited for all of the keywords except `set_color`. The scope for `set_color` begins when the `add_grid` keyword is encountered, and continues until a part, wheel, or reference frame is added.

Target Path

The animator can show target paths specified as inputs to the driver model in CarSimEd. There are three kinds of input recognized by the animator: the color of the path, a design path, and an offset to the design path. The target path for the controller is the combination of the design path plus the offset, as described in the Chapter 9 section **Input: Target Path For Closed-Loop Steer Control**. The target path (including the lateral offset) is drawn by the animator with dashed lines, and the design path (without the lateral offset) is drawn with solid lines.

Table E.6. Keywords for describing parts.

Keyword	**Value**	**Description**
set_path_color	color (9 choices): black, white, red, blue, green, magenta, yellow, dark gray, light gray	color used for design and target paths
yin_table endtable	list of X-Y coordinates: 2 numbers per line	coordinates of design path
ltarg_table endtable	list of S-L coordinates: 2 numbers per line	lateral offset of the target path from the design path (S=distance along design path, L=lateral offset to the left)

The scope of the target path inputs is unlimited. The three keywords shown in Table E.6 can appear anywhere in the input PAR files.

The ltarg_table input is not used in CarSimEd (it is present in the commercial CarSim and TruckSim packages that use the same animator program).

Appendix F — Model Files and Keywords

This appendix describes the input and output files associated with the solver programs in CarSimEd. It also lists an example echo file with the keywords recognized by the CarSimEd solver programs.

File Types

Every time a new run is made from the **Runs** screen, the seven files listed in Table F.1 are created.

Table F.1. Standard files created when a simulation is run.

Name	Location	Creator	Description
`Simfile`	`Programs`	SGUI	Batch control file.
<*id*>.PAR	`Runs`	SGUI	List of parameters, read as input by program.
<*id*>.LPO	`Runs`	Program	List of parameters, written as output by program. Also contains initial conditions. Can be used to repeat a run.
<*id*>.LPF	`Runs`	Program	List of parameters and final conditions, written as output by program. Can be used to continue a run.
<*id*>.ERD	`Runs`	Program	Header for ERD file, needed by plotter and animator programs.
<*id*>.BIN	`Runs`	Program	Numerical values of output variables, stored in binary form.
<*id*>.LOG	`Runs`	Program	List of all PARSFILEs that were processed to make a run.

One file always has the same name—`Simfile`. The other six all have the same base name— the current ID number displayed in the upper-corner of the **Runs** screen.

Simfile

`Simfile` is a batch control file. When a solver program starts, it looks for `Simfile`. If the file is not found, the program will prompt you for an input file name. If `Simfile` is found, the program obtains all necessary input and output file names from it. An example `Simfile` is shown below.

Listing F.1. Example Simfile.

```
SIMFILE
INPUT C:\CARSIMED.45\RUNS\575.PAR
ECHO C:\CARSIMED.45\RUNS\575.LPO
FINAL C:\CARSIMED.45\RUNS\575.LPF
ERDFILE C:\CARSIMED.45\RUNS\575.ERD
LOGFILE C:\CARSIMED.45\RUNS\575.LOG
END
```

Although it is a convention in CarSimEd to use the same root name for all files, the individual solver programs are capable of creating any names allowable under the file operating system.

Notice that the names in `Simfile` are full pathnames. The individual files can be located anywhere on the computer or a network where volumes are accessed using the DOS convention.

Whenever you click the **Run** button, a new `Simfile` is created in the same folder as the solver program (by convention, this is the folder `Programs`). If a file with the name `simfile` already exists, it is overwritten.

PAR Files

The PAR file is read by the solver program and provides values of model parameters for the forthcoming run.

Listing F.2. Example PAR file.

```
PARSFILE
parsfile C:\CARSIMED.45\VEHICLES\VEHICLES\58.par
parsfile C:\CARSIMED.45\INPUT\STEERING\27.par
parsfile C:\CARSIMED.45\INPUT\BRAKING\75.par
parsfile C:\CARSIMED.45\INPUT\FRICTION\23.par
parsfile C:\CARSIMED.45\COMP_PAR\41.par
stopt 2
speed 80

* used by animator program
PARSFILE C:\CARSIMED.45\ANIMATE\CAMERAS\244.par
* used by plotting programs
PARSFILE C:\CARSIMED.45\RUNS\751.plt

testid 751
title Base test

END
```

When you click the **Run** button, a new PAR file is created in the folder containing the Runs library (i.e., the `Runs`, which contains Runs`.tbk` file). The PAR file contains information from the **Runs** screen. As shown in the following example, the file is relatively short.

Each link seen in the **Runs** screen is represented with a full pathname to an existing data file. In the above example, the data from the selected vehicle is contained in the file `58.PAR`, in the directory `C:\CARSIMED.45\VEHICLES\VEHICLES`.

The keyword PARSFILE is similar to the INCLUDE directive in C or FORTRAN: it instructs the program to open the specified file and to continue reading from the new file. When the new file has been scanned to its end, the program continues reading from the original file.

PARSFILEs are often nested five or six layers deep.

In order to view all of the parameters used in a simulation run, you could trace the input files the same way as the program: start with the Simfile to find the name of the input (PAR) file. Then, open the input file and view any parameter values in it. Open any files identified with the keyword PARSFILE, and continue the inspection.

A program called Parstree is included in CarSimEd to rapidly inspect the "tree" of PARSFILEs. To use it, just click the Tree button in the ribbon bar: . (See Chapter 5 for more information.)

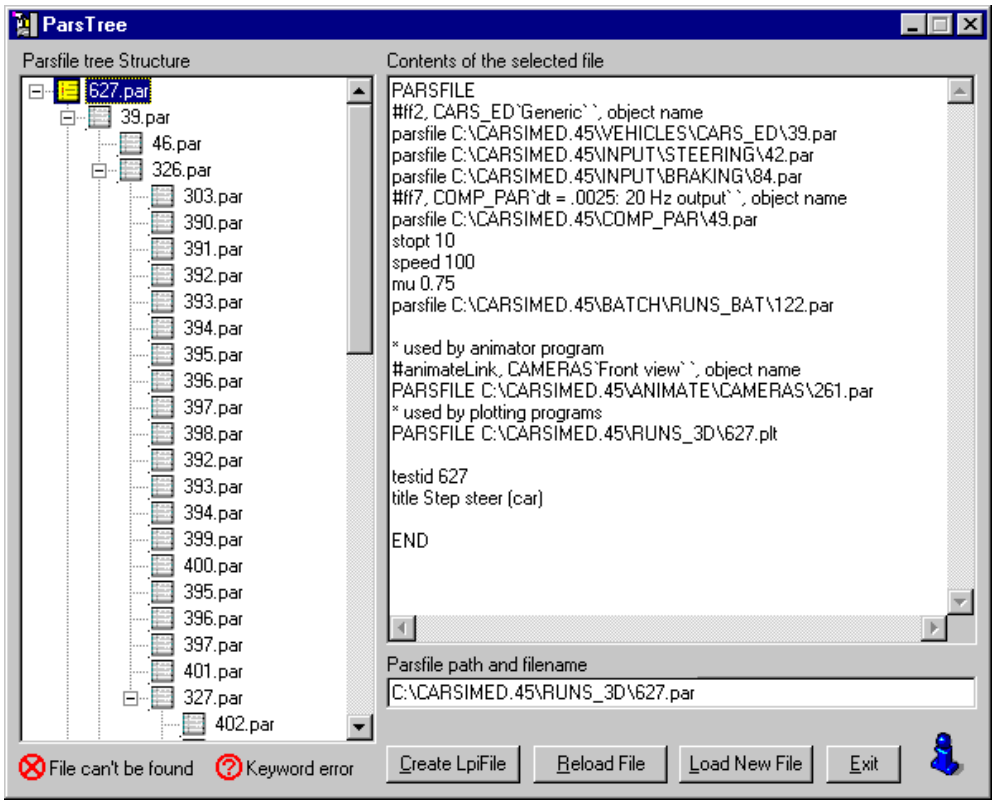


Figure F.1. Display of nested PARSFILEs with Parstree.

However, there are faster ways to get the information. The list of all files is contained in the LOG file, and the list of all parameter values is contained in both the LPO and LPF files.

LPO and LPF Echo Files

In CarSimEd, the input parameter values are typically spread over many files. When a solver program runs, it creates summary files that list each and every parameter value that is being used in the current simulation. These even include parameters that were not given values (in these cases the default values are listed).

One of these files is created before the run (LPO), and the other is created at the end of the run (LPF). Both contain all parameter values used in the simulation. In addition, the LPO file contains the initial conditions of the state variables in the simulation model. The information in the LPO file is sufficient to exactly repeat a run, using only the LPO file as an input.

An example LPO file is listed later, to show all of the keywords recognized by a CarSimEd model.

The LPF file is nearly identical, except that instead of the initial values, it contains the final values of the state variables. By modifying the start and stop time parameters, an existing run can be continued if the LPF file is used as the input file for the simulation solver program.

ERD and BIN Files

Generally, the main purpose of each solver program is to predict time histories of variables of interest. These time histories are stored in a binary data file with the extension BIN. A companion file, with extension ERD, describes the layout of the BIN file and also contains labeling information for each variable.

The format of an ERD file is described in Appendix C. Appendix J contains a list of all of the variables contained in the ERD and BIN files for one of the CarSimEd models.

LOG Files

The input parameters for the simulation models are spread over many screens in the CarSimEd database. Each data set is written in a PARSFILE. When the solver program reads the inputs, it writes a record of every PARSFILE into a LOG file. This is mainly used for debugging—most users will never have occasion to look at LOG files.

When the solver program executes, it reads the PARSFILE specified in the SIMFILE. If that PARSFILE includes the keyword PARSFILE, the named file is opened and the program continues reading from the newly opened file. For example, the top-level file shown in Listing F.2 contains PARSFILE keywords that identify other files with more specific types of data. Vehicle parameters are obtained by reading the file `C:\CARSIMED.45\VEHICLES\CARS\58.par`; steering input data are obtained by reading the file parsfile `C:\CARSIMED.45\INPUT\STEERING\27.par`; a braking input is obtained by reading the file `C:\CARSIMED.45\INPUT\BRAKING\75.par`;

and so on. When the vehicle file is read, the program finds more references to other PARSFILEs, with data for suspensions, tires, and various vehicle properties.

The LOG file is a list of every PARSFILE that was referenced, starting with the file named in the SIMFILE. Listing F.3 shows the top portion of a LOG file.

Listing F.3. Example LOG File.

```
Include PARSFILE "C:\CARSIMED.45\RUNS\603.PAR"
Include PARSFILE "C:\CARSIMED.45\VEHICLES\VEHICLES\58.par"
Include PARSFILE "C:\CARSIMED.45\VEHICLES\AERO\34.par"
 Example aero data (full)
Include PARSFILE "C:\CARSIMED.45\VEHICLES\STEER\STRS_CAR\40.par"
Include PARSFILE "C:\CARSIMED.45\VEHICLES\STEER\STR_4W\12.par"
 Front Wheel Steer
...
```

Viewing Keywords

The files read by the solver programs in CarSimEd are scanned for input data. Parameters are identified by keywords.

Viewing a Complete List of Parameters

The list of parameters in the next section was created from within the software for a particular run. To create a similar list for other runs, do the following:

1. Go to the **Runs** library, and find a run involving the vehicle of interest.
2. If the run has not already been made, click the **Run** button.
3. Click the **View All Parameters** button to view the file in a text editor.

Alternatively, look at the ID number in the upper-right corner of the **Runs** screen (e.g., 430). Then, use a file browser such as Windows Explorer to find the files with the ID numbers as a name with extensions LPO and LPF (e.g., `430.LPO`, `430.LPF`). Either one contains all keywords recognized by the solver program.

Indexed Keywords

Many of the parameters in the CarSimEd models are applied in several places on the vehicle. For example, each tire has a similar set of parameters. The same data screen is used to describe a tire whether it is on a left-front or right-rear wheel. The location of the wheel is specified with the additional keywords `IAXLE` and `ISIDE`.

Parameter values such as an axle mass are written in the PARSFILE with numerical indices, e.g., `MUS(1)` indicates the unsprung mass of axle 1. Tabular data are written with keywords in front of the table to indicate which axle, side, etc. are associated with the table.

Keywords and Parameters for the 3D Car Model

CarSimEd has two separate programs for solving the equations of motion of the 3D car model: one is a stand-alone program (the file extension is EXE) and the other is a dynamic link library (the file extension is DLL) for use with MATLAB/SIMULINK.

Both have the same equations of motion, and use the same keywords to identify parameters and tables.

```
PARSFILE
* CarSimEd 3D vehicle.
* Generated by AutoSim 2.81 (PPC Dev) on January 20, 2000.
* Copyright 1996-2000. Mechanical Simulation Corporation.
* All rights reserved.

TITLE Step steer (car)

* Input File: C:\CARSIMED.45\RUNS_3D\627.PAR
* Run was made 12:23 on Jan. 20, 2000
FORMAT BINARY

IPRINT          20 , number of time steps between output printing (counts)
STARTT           0 , simulation start time (s)
STEP        0.0025 , simulation time step (s)
STOPT           10 , simulation stop time (s)

* PARAMETER VALUES

ASW_MAX         360 , Maximum allowed steering wheel angle (in driver model) (deg)
CSFY(1)     -0.0003 , Front suspension compliance: d(steer)/d(Fy) (deg/N)
CSFY(2)           0 , Rear suspension compliance: d(steer)/d(Fy) (deg/N)
CSMZ(1)       0.004 , Front suspension compliance: d(steer)/d(Mz) (deg/N/m)
CSMZ(2)       0.002 , Rear suspension compliance: d(steer)/d(Mz) (deg/N/m)
CSMZF         0.001 , Steering system compliance: d(steer)/d(Mzl + Mzr)/2) (deg/N/m)
CTFX(1)      0.0004 , Front suspension compliance: d(toe)/d(Fx) (deg/N)
CTFX(2)           0 , Rear suspension compliance: d(toe)/d(Fx) (deg/N)
DS(1)          0.75 , Front damper rate, at shock absorber (N-s/mm)
DS(2)          0.75 , Rear damper rate, at shock absorber (N-s/mm)
HCG             550 , Nominal height of entire vehicle C.G. (mm)
HRC(1)          100 , Nominal height of front axle roll center (mm)
HRC(2)          100 , Nominal height of rear axle roll center (mm)
HWC(1)          285 , Undeflected height of front wheel center (mm)
HWC(2)          285 , Undeflected height of rear wheel center (mm)
IW(1)           1.1 , Spin moment of inertia of front wheel (kg-m2)
IW(2)           1.1 , Spin moment of inertia of rear wheel (kg-m2)
IXX             400 , Moment of inertia of entire vehicle (kg-m2)
IXZ               0 , Product of inertia of entire vehicle (kg-m2)
```

```
IYY             2704 , Moment of inertia of entire vehicle (kg-m2)
IZZ             3136 , Moment of inertia of entire vehicle (kg-m2)
KAUX(1)          500 , Front auxiliary stiffness, including anti-sway bar (N-m/deg)
KAUX(2)          350 , Rear auxiliary stiffness, including anti-sway bar (N-m/deg)
KFX(1)         100000 , Front tire longitudinal stiffness (N)
KFX(2)         100000 , Rear tire longitudinal stiffness (N)
KFYCAM(1)         -60 , Front tire camber stiffness (N/deg)
KFYCAM(2)         -60 , Rear tire camber stiffness (N/deg)
KS(1)              30 , Front suspension spring stiffness (at spring) (N/mm)
KS(2)              20 , Rear suspension spring stiffness (at spring) (N/mm)
KT(1)             200 , Front tire vertical stiffness (N/mm)
KT(2)             200 , Rear tire vertical stiffness (N/mm)
LCGT          1111.76 , CALC - Distance from F axle to total vehicle CG (mm)
LRELAX(1)          600 , Front tire relaxation length (mm)
LRELAX(2)          600 , Rear tire relaxation length (mm)
LTK(1)            1500 , Front axle track width (mm)
LTK(2)            1500 , Rear axle track width (mm)
LWB               2700 , Wheelbase (mm)
MF                1000 , Vehicle mass supported by front axle (2 wheels) (kg)
MR                 700 , Vehicle mass supported by rear axle (2 wheels) (kg)
MT                1700 , CALC - Total vehicle mass (kg)
MU                0.75 , Tire/ground friction coefficient (-)
MUS(1)             100 , Front axle unsprung mass (2 wheels) (kg)
MUS(2)              80 , Rear axle unsprung mass (2 wheels) (kg)
RAP(1)             0.1 , Wheelbase change per unit jounce at front axle (-)
RAP(2)             0.1 , Wheelbase change per unit jounce at rear axle (-)
RCAM(1)              0 , Wheel camber change per unit jounce at front axle (deg/mm)
RCAM(2)              0 , Wheel camber change per unit jounce at rear axle (deg/mm)
RDAMP(1)             1 , Front ratio of jounce at wheel to damper stroke (-)
RDAMP(2)             1 , Rear ratio of jounce at wheel to damper stroke (-)
RMF           0.588235 , CALC - Ratio: proportion of load on front axle (-)
RMR           0.411765 , CALC - Ratio: proportion of load on rear axle (-)
RMYBK(1)            10 , Front wheel ratio of brake torque to pedal input (N-m/MPa)
RMYBK(2)             4 , Rear wheel ratio of brake torque to pedal input (N-m/MPa)
RMYTH(1)           500 , Front wheel ratio of drive torque to throttle input (N-m)
RMYTH(2)             0 , Rear wheel ratio of drive torque to throttle input (N-m)
ROLL_STOP           45 , Roll angle for stopping the simulation (deg)
RSPRNG(1)            1 , Front ratio of suspension jounce to spring compression (-)
RSPRNG(2)            1 , Rear ratio of suspension jounce to spring compression (-)
RSW                 16 , Steering gear ratio (-)
RTIME              0.1 , CALC -- Computational efficiency (sec/sim. sec) (-)
RTOE(1)         -0.006 , Wheel toe change per unit jounce at front axle (deg/mm)
RTOE(2)          0.006 , Wheel toe change per unit jounce at rear axle (deg/mm)
SPEED              100 , Vehicle forward speed (kph)
SPEED_ON_OFF         1 , Speed control switch  (0.0 -> off, 1.0 -> on) (-)
STARTS         219.798 , Starting station number (beginning of simulation) (m)
```

```
STOPS          10000 , Stopping station number (stop simulation when this is reached) (m)
TDLAG              0 , Lag time used by driver model (s)
TPREV              1 , Preview time used by driver model (s)
VLOW_ALPHA(1)          5 , Front low-speed threshold for modified tire relaxation equations
(kph)
VLOW_ALPHA(2)          5 , Rear low-speed threshold for modified tire relaxation equations
(kph)
VLOW_KAPPA(1)          2 , Front low-speed threshold for modified longitudinal slip equations
(kph)
VLOW_KAPPA(2)          2 , Rear low-speed threshold for modified longitudinal slip equations
(kph)
VLOW_SPINA(1)          2 , Front low-speed threshold for modified wheel spin equations (kph)
VLOW_SPINA(2)          2 , Rear low-speed threshold for modified wheel spin  equations (kph)
V_STOP            -1 , Low-speed limit for stopping the simulation (kph)
XDESIGN       219.798 , NIL (-)
YDESIGN       142.078 , NIL (-)

* Brake input vs. time
BRKIN_TABLE
         0,          0
        10,          0
ENDTABLE


* Cornering stiffness vs. load (30 pts max)
IAXLE     1 , Table ID number
KFYA_TABLE
         0,          0
       500,        181
      1000,        356
      2000,        673
      3000,        922
      4000,       1097
      6000,       1253
      8000,       1250
ENDTABLE

IAXLE     2 , Table ID number
KFYA_TABLE
         0,          0
       500,        181
      1000,        356
      2000,        673
      3000,        922
      4000,       1097
      6000,       1253
```

```
      8000,       1250
ENDTABLE


* Steering wheel input vs. time. Column 1 = time (sec). Column 2 =
* steering wheel angle (deg).
STEERSW_TABLE
         0,         0
       0.1,        30
         3,        30
ENDTABLE


* Throttle input vs. time
THROTTLE_TABLE
         0,         0
         2,         0
ENDTABLE


* Pneumatic trail vs. load (30 pts max)
IAXLE     1 , Table ID number
TRAIL_TABLE
         0,        12
      2000,        15
      4000,        23
      6000,        36
      8000,        52
ENDTABLE

IAXLE     2 , Table ID number
TRAIL_TABLE
         0,        12
      2000,        15
      4000,        23
      6000,        36
      8000,        52
ENDTABLE


* FINAL CONDITIONS

XO             219.8114951976 , Abs. X trans. of S0 (m)
YO             142.1095052499 , Abs. Y trans. of S0 (m)
ZO          0.003076073890054 , Abs. Z trans. of S0 (m)
YAW             67.51149098427 , Abs. Z rot. of Spp (deg)
```

```
PITCH        -0.00224071736904 , Y rot. of Sp rel. to Spp (deg)
ROLL           1.963382153352 , X rot. of S rel. to Sp (deg)
JNC_WLF      -0.02247700439532 , Trans. of WLF0 rel. to WLFJ (m)
JNC_WLR       -0.0242091532937 , Trans. of WLR0 rel. to WLRJ (m)
JNC_WLR       0.01650853050917 , Trans. of WRF0 rel. to WRFJ (m)
JNC_WRR       0.01828318644881 , Trans. of WRR0 rel. to WRRJ (m)
ROTLF          55688.11428731 , Rotation angle of LF wheel (deg)
ROTLR          55628.85102808 , Rotation angle of LR wheel (deg)
ROTRF          56038.73084637 , Rotation angle of RF wheel (deg)
ROTRR          55983.15443978 , Rotation angle of RR wheel (deg)
TANLF        -0.03409684625009 , Slip angle tan of LF tire, with time lag (-)
TANRF        -0.03168883381219 , Slip angle tan of RF tire, with time lag (-)
TANLR        -0.02750321007954 , Slip angle tan of LR tire, with time lag (-)
TANRR        -0.02625441847659 , Slip angle tan of RR tire, with time lag (-)
IVERR          0.181071087701 , Integral of velocity error for speed controller (m)
VXS           27.77553913376 , Vehicle forward speed (m/s)
VYS          -0.514535513897 , Abs. Y trans. speed of SCMC (m/s)
VZS         0.01654330461462 , Abs. Z trans. speed of SCMC (m/s)
AVZ           6.771926099055 , Abs. Z rot. speed of S (deg/s)
AVY          0.2306196465965 , Abs. Y rot. speed of S (deg/s)
AVX       -0.0004862471536275 , Abs. X rot. speed of S (deg/s)
JNCR_WLF   -9.358694147682e-006 , Trans. speed of WLF0 rel. to S (m/s)
JNCR_WLR    5.518184953048e-005 , Trans. speed of WLR0 rel. to S (m/s)
JNCR_WLR   -2.486989249608e-005 , Trans. speed of WRF0 rel. to S (m/s)
JNCR_WRR    3.936770925572e-005 , Trans. speed of WRR0 rel. to S (m/s)
AVY_LF         15.47666753711 , Spin rate of LF wheel (rev/s)
AVY_LR          15.4610040621 , Spin rate of LR wheel (rev/s)
AVY_RF         15.57431795697 , Spin rate of RF wheel (rev/s)
AVY_RR         15.56005195726 , Spin rate of RR wheel (rev/s)

END
```

Keywords and Parameters for the 2D Car Model

```
PARSFILE
* 2D Simple ride model dynamic simulation.
* Version created by AutoSim 2.8 on February 26, 1999.
* Copyright 1999. Mechanical Simulation Corporation. All rights reserved.

TITLE Sine Sweep

* Input File: C:\CARSIMED\RUNS_2D\640.PAR
* Run was made 17:31 on Jul. 23, 1999
FORMAT BINARY
```

```
IPRINT          1 , number of time steps between output printing (counts)
STARTT          0 , simulation start time (s)
STEP       0.0025 , simulation time step (s)
STOPT          13 , simulation stop time (s)

* PARAMETER VALUES

DS(1)        0.75 , Front damper rate, at shock absorber (N-s/mm)
DS(2)        0.75 , Rear damper rate, at shock absorber (N-s/mm)
HWC(1)        285 , Height of front wheel center (used only for animation) (mm)
HWC(2)        285 , Height of front wheel center (used only for animation) (mm)
IYY          2704 , Moment of inertia of entire vehicle (kg-m2)
KS(1)          30 , Front suspension spring stiffness (at spring) (N/mm)
KS(2)          20 , Rear suspension spring stiffness (at spring) (N/mm)
KT(1)         200 , Front tire vertical stiffness (N/mm)
KT(2)         200 , Rear tire vertical stiffness (N/mm)
LTK(1)       1500 , Front axle track width (used only for animation) (mm)
LTK(2)       1500 , Rear axle track width (used only for animation) (mm)
LWB          2700 , Wheelbase (mm)
MF           1000 , Vehicle mass supported by front axle (2 wheels) (kg)
MR            700 , Vehicle mass supported by rear axle (2 wheels) (kg)
MUS(1)        100 , Front axle unsprung mass (2 wheels) (kg)
MUS(2)         80 , Rear axle unsprung mass (2 wheels) (kg)
RDAMP(1)        1 , Front ratio of damper stroke to suspension movement (-)
RDAMP(2)        1 , Rear ratio of damper stroke to suspension movement (-)
RSPRNG(1)       1 , Front ratio of spring compression to suspension movement (-)
RSPRNG(2)       1 , Rear ratio of spring compression to suspension movement (-)
RTIME           0 , CALC -- Computational efficiency (sec/sim. sec) (-)
SPEED          72 , Vehicle forward speed (km/h)

* Road profile (Z coord. vs. X coord.)
ROAD_PROFILE_TABLE
         0,         0
       1.5,         0
      2.99,     0.002
…
     259.4,    -0.043
ENDTABLE


* FINAL CONDITIONS

XO            260.0313110352 , Abs. X trans. of S0 (m)
ZO       -0.003454771125689 , Abs. Z trans. of S0 (m)
PITCH     -0.02440359257162 , Abs. Y rot. of S (deg)
JNC_WLF    -0.01780941709876 , Z trans. of WLF0 rel. to S0 (m)
```

```
JNC_WLR       0.04939780384302 , Z trans. of WLR0 rel. to WLRJ (m)
VZS          -0.07862732559443 , Abs. Z trans. speed of SCMC (m/s)
AVY           4.837102413177 , Abs. Y rot. speed of S (deg/s)
JNCR_WLF      0.9013347029686 , Z trans. speed of WLF0 rel. to S (m/s)
JNCR_WLR       1.783175468445 , Z trans. speed of WLR0 rel. to S (m/s)

END
```

Keywords and Parameters for the 3D Suspension

```
PARSFILE
* Echo file created by:
* Kinematic simulation of 5-point suspension.
* Version created by AutoSim 2.58 on February 6, 1997.
* Copyright 1989-1996 The Regents of The University of Michigan. All rights reserved.

TITLE Suspension

* Input File: C:\CARSIMED.45\RUNS_SUS\641.PAR
* Run was made 12:11 on Jan. 21, 19100
CFORMAT binary

IPRINT          1 , number of time steps between output printing (counts)
STARTT          0 , simulation start time (s)
STEP         0.01 , simulation time step (s)
STOPT           1 , simulation stop time (s)

* PARAMETER VALUES

BSY           690 , Y coordinate of point on wheel spin axis (mm)
BSZ            -1 , Z coordinate of point on wheel spin axis (mm)
BY            770 , Y coordinate of wheel center (mm)

BZ_TABLE ht of point B vs. time
        0,      -220 , point in table: (s, mm)
        1,       220 , point in table: (s, mm)
ENDTABLE

PX(1)          -60 , X coordinate of chassis point (mm)
PX(2)         -300 , X coordinate of chassis point (mm)
PX(3)          -90 , X coordinate of chassis point (mm)
PX(4)         -240 , X coordinate of chassis point (mm)
PX(5)          210 , X coordinate of chassis point (mm)
PY(1)          410 , Y coordinate of chassis point (mm)
PY(2)          430 , Y coordinate of chassis point (mm)
```

```
PY(3)          370 , Y coordinate of chassis point (mm)
PY(4)          390 , Y coordinate of chassis point (mm)
PY(5)          380 , Y coordinate of chassis point (mm)
PZ(1)          330 , Z coordinate of chassis point (mm)
PZ(2)          290 , Z coordinate of chassis point (mm)
PZ(3)            0 , Z coordinate of chassis point (mm)
PZ(4)         -110 , Z coordinate of chassis point (mm)
PZ(5)         -100 , Z coordinate of chassis point (mm)
XX(1)          -60 , X coordinate of carrier point (mm)
XX(2)         -120 , X coordinate of carrier point (mm)
XX(3)         -190 , X coordinate of carrier point (mm)
XX(4)            0 , X coordinate of carrier point (mm)
XX(5)            0 , X coordinate of carrier point (mm)
XY(1)          640 , Y coordinate of carrier point (mm)
XY(2)          640 , Y coordinate of carrier point (mm)
XY(3)          650 , Y coordinate of carrier point (mm)
XY(4)          740 , Y coordinate of carrier point (mm)
XY(5)          740 , Y coordinate of carrier point (mm)
XZ(1)          350 , Z coordinate of carrier point (mm)
XZ(2)          340 , Z coordinate of carrier point (mm)
XZ(3)          -20 , Z coordinate of carrier point (mm)
XZ(4)         -130 , Z coordinate of carrier point (mm)
XZ(5)         -130 , Z coordinate of carrier point (mm)

* FINAL CONDITIONS

Q(1)         60.3664 , CALC--Abs. X rot. of R1p (deg)
Q(2)         11.1251 , CALC--Z rot. of R1 rel. to R1p (deg)
Q(3)        -50.0164 , CALC--X rot. of WCpp rel. to R1 (deg)
Q(4)         4.45457 , CALC--Y rot. of WCp rel. to WCpp (deg)
Q(5)        -9.65248 , CALC--Z rot. of WC rel. to WCp (deg)
Q(6)         59.4151 , CALC--Abs. X rot. of R2p (deg)
Q(7)         11.0191 , CALC--Z rot. of R2 rel. to R2p (deg)
Q(8)         41.6627 , CALC--Abs. X rot. of R3p (deg)
Q(9)         3.55033 , CALC--Z rot. of R3 rel. to R3p (deg)
Q(10)         36.9715 , CALC--Abs. X rot. of R4p (deg)
Q(11)        0.983953 , CALC--Z rot. of R4 rel. to R4p (deg)
Q(12)         36.3276 , CALC--Abs. X rot. of R5p (deg)
Q(13)        0.967013 , CALC--Z rot. of R5 rel. to R5p (deg)

END
```

Appendix G — The 3D Car Model

This appendix describes the modeling assumptions that were used to build the 3D car model in CarSimEd. The tire model is described in Appendix H. Portions of both appendices have been published previously [1] (numbers in brackets refer to documents listed at the end of this appendix).

Introduction

It is often said that an automobile is controlled by forces developed in just four small patches, each the size of a man's hand, where the tires contact the road. In the 1940's and 1950's, researchers such as Lanchester, Olley, Rieckert and Schunk, Rocard, and Segel developed an understanding of how tire forces are generated and affect the steering and braking behavior of the vehicle [2]. Segel and other early researchers in the 1950's developed linear equations by hand, and solved them using frequency-domain analysis [3]. Segel's classic model reduced the vehicle behavior to its essence, with a minimal number of parameters and variables and just three degrees of freedom (DOF).

From the 1960's to the early 1980's, the proliferation and improvement of analog and then digital computers led to a new phase of vehicle modeling, in which many automotive simulation programs were developed and refined by research engineers. The new computer models were more complex, typically with 10 to 20 DOF [4, 5]. The additional complexity accounted for nonlinearity and more detailed suspension kinematics. Equations were still formulated by hand, and coded by hand in computer language for numerical solution in specialized programs. The more detailed models involved many years of development, not counting the efforts spent in validation and verification.

Starting with the mid-1980's, engineers started using newly available multibody simulation programs to describe the model geometrically, "assembling" the system model from components [6, 7]. Modelers no longer had to derive equations, and therefore, the efforts and potential errors associated with deriving equations and coding them were nearly eliminated.

Automotive manufacturers and many others now use multibody programs to perform simulations of automotive handling and braking behavior [8]. The tendency has been to include nearly all moving parts in the suspensions and steering systems. Inputs include coordinates of most joints between parts, and mass properties of individual parts. In contrast, the earlier custom programs were more systems-oriented, involving generalized movements of wheels relative to the body, or, even more simply, movements of the body relative to the ground. The advantage of the detailed multibody programs for development engineers is that they can fine-tune designs by modifying component-level details. However, the detailed models also have some disadvantages. Engineers who do

not work for car manufacturers may not have access to the geometric design data. Even when the full set of input parameters is assembled, the programs run slower than custom programs that are less complex. (With some multibody programs, the run-time performance is much slower even for comparable models. For models with a complexity similar to the one presented in this appendix, a numerical multibody programs might be a couple orders of magnitude slower than a hand-written program specialized for a specific vehicle dynamics model.)

The creation of CarSimEd was motivated in part by the thought that something has been lost during the evolution from the older models to the newer. The insight and expertise that underlay the old hand-written models are often lacking in modern multibody models. Although the modern models are often highly detailed, their accuracy in predicting vehicle response to steering and braking inputs is sometimes not as good as that obtained 40 years ago. This appendix is intended to convey some of the ideas and concepts used in earlier vehicle models for applications using modern multibody programs.

The model that will be described was implemented using the AutoSim multibody code generator [8, 9]. AutoSim generates equations symbolically, performs coding optimizations, and generates a custom simulation program. The simulation program obtained by AutoSim has run-time performance comparable with (and usually better than) that of a hand-coded program based on the same model.

Overview of Factors Affecting Vehicle Behavior

Figure G.1 shows a free-body diagram of a four-wheeled vehicle as viewed from the top. There are just three governing equations: the sum of the tire shear forces must equal the vehicle mass times its acceleration in both the vehicle X and Y directions, and the moment of those forces about the vehicle mass center must be equal to the product of the yaw acceleration and the vehicle yaw moment of inertia. Thus, the main objective of the vehicle model is to accurately predict tire shear forces.

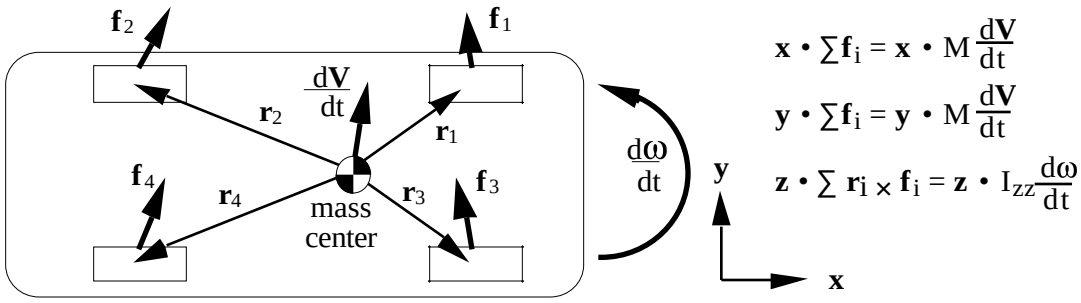


Figure G.1. Primary factors influencing vehicle system motions.

A vehicle is also subject to aligning moments in the tire contact patches. The aligning moment has a negligible direct effect on the vehicle yaw, but, due to steering compliance,

it can have a significant influence in determining the all-important shear forces. Another behavior that influences the vehicle response involves the rotary motion of the car body in roll and pitch. Mechanical energy is transfered to the sprung mass as the vehicle pitches and rolls, and these motions contribute to the vehicle transient response.

Besides the tire/road interactions, the only forces and moments acting on the vehicle are due to aerodynamic effects. They have a secondary influence, but are relatively easy to add to multibody models.

Rigid Body Kinematics

The model is based on a rigid body that represents the main body of the vehicle and has six DOF. For the vehicle with front and rear independent suspension, an additional four bodies are added, each with a single translational DOF, to account for the vertical movements allowed by the suspensions. The wheel bodies are positioned such that the origins of their local coordinate systems are nominally at the locations of the centers of tire contact (see Figure G.2). Longitudinally, the origins of the front and rear wheels are separated by the vehicle wheelbase, L_{wb}. Laterally, they are separated by the vehicle front and rear track widths, $L_{tk,f}$ and $L_{tk,r}$.

If the influence of compliances in the suspension and steering system linkages are removed, each wheel center of a real vehicle follows a trajectory through 3D space, relative to the car body, as the suspension moves up and down. Due to the kinematics of the suspension, the trajectory is usually not purely vertical. For most vehicles, the wheels move out laterally as the suspensions are compressed, such that track width increases with suspension compression. The wheels also move out longitudinally, such that the wheelbase increases with suspension compression.

The direction of the wheel trajectory (relative to the main body) determines how tire shear forces in the ground plane are transmitted to the vehicle body through reaction forces in the suspension linkages. In hand-written equations, roll and pitch moments due to suspension reaction forces have been written with coefficients with names such as anti-roll, anti-pitch, anti-dive, anti-squat, and jacking [10].

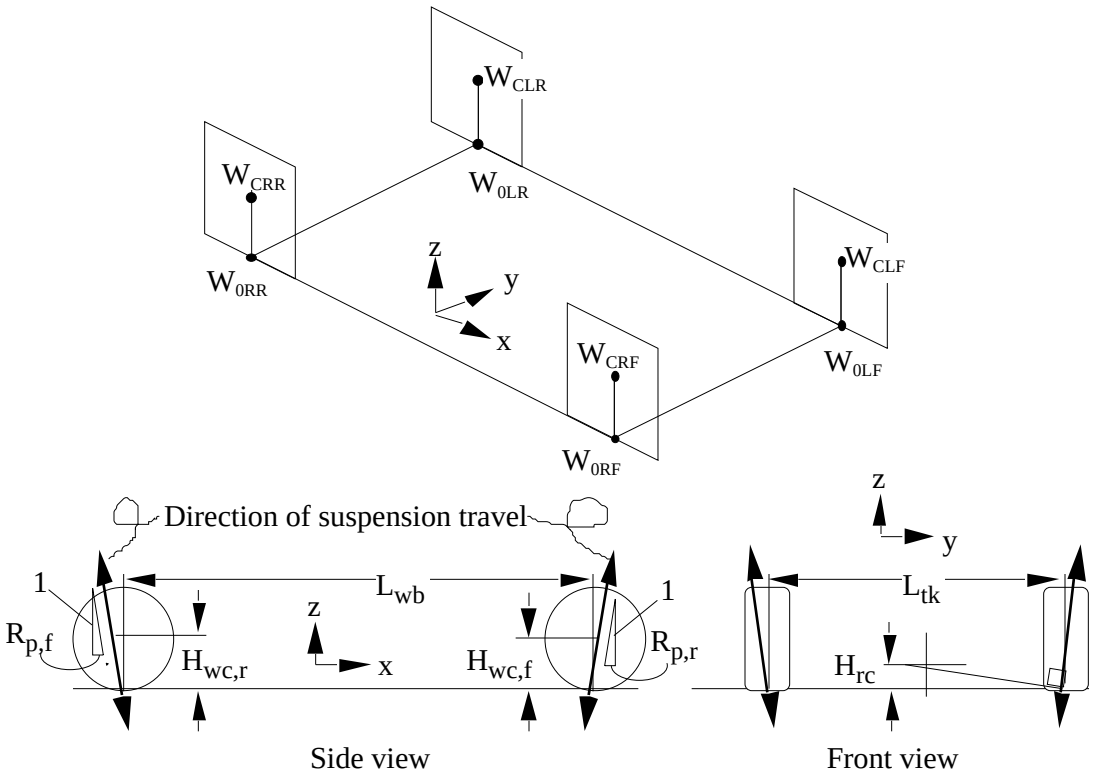


Figure G.2. Locations and movements of wheels.

The multibody model accounts for the interaction between tire shear forces and roll and pitch moments so long as the movement is constrained to follow the proper path. A simple approximation is to assume the movement is in a straight line, as shown in Figure G.2. Using an axis system based in the vehicle sprung mass ($\mathbf{s}_x$, $\mathbf{s}_y$, $\mathbf{s}_z$), the directions of the movements of the four wheels are:

$$\text{left-front: } \text{dir}\left(\mathbf{s}_z + \frac{2H_{rc,f}}{L_{tk,f}}\,\mathbf{s}_y + R_{p,f}\,\mathbf{s}_x\right)$$

$$\text{right-front: } \text{dir}\left(\mathbf{s}_z - \frac{2H_{rc,f}}{L_{tk,f}}\,\mathbf{s}_y + R_{p,f}\,\mathbf{s}_x\right)$$

$$\text{left-rear: } \text{dir}\left(\mathbf{s}_z + \frac{2H_{rc,r}}{L_{tk,r}}\,\mathbf{s}_y - R_{p,r}\,\mathbf{s}_x\right)$$

$$\text{right-rear: } \text{dir}\left(\mathbf{s}_z - \frac{2H_{rc,r}}{L_{tk,r}}\,\mathbf{s}_y - R_{p,r}\,\mathbf{s}_x\right) \qquad (1)$$

where dir is a function that returns the direction of a vector, and the secondary subscripts f and r indicate parameters for the front and rear.

In the traditional suspension analyses, the roll kinematics are often analyzed to define a point called a roll center [10]. The independent suspension model in CarSimEd does not use the roll-center concept. However, for compatibility with this convention, the

inclination of the wheel movement in the roll direction is defined by the ratio of a roll center height to the half-track distance.

> Note: The existence of a roll-center parameter should not be interpreted to mean the CarSimEd model has a roll center. As with real vehicles, the actual roll center moves in the CarSimEd model once it is out of the equilibrium condition. Given that the roll center is not used in the model, and that it does not correspond to a physical part of a real vehicle, a dynamic analysis to calculate a roll center was not included in the CarSimEd models.

A single coefficient (R_p) is used to define the longitudinal inclination of the wheel movement.

Masses and inertias

The user of a vehicle model must provide mass and inertia parameters for the bodies in the model. In CarSimEd the moments of inertia of the wheel bodies are defined as zero. The mass centers are located at the wheel centers, nominally a height H_{wc} above the ground. The mass of each wheel body should be set to that portion of vehicle mass supported by the tire that is considered to move with the wheel. This value is commonly called the unsprung mass, and usually includes some of the mass of the suspension elements.

The mass of the main body (the sprung mass) is set to the mass of the entire vehicle minus the unsprung masses. The inertia properties are also required, including the XZ product of inertia. (Due to lateral symmetry, the XY and YZ products are defined as zero.) It is much easier to measure inertia properties for the entire vehicle than for the body alone. The multibody program can be made to calculate the mass and inertia properties of the sprung mass from measurements made for the entire vehicle. This is done by adding four more bodies and giving them negative masses. These four bodies should be placed at the same locations as the wheel body mass centers. However, their masses are set to the negative values of the unsprung masses, and they are fully constrained with respect to the main vehicle body (i.e., zero DOF). The multibody program, in accounting for the full constraint of these four bodies, will in effect subtract the masses and inertia properties, bringing the mass and inertias of the main body down to those of the sprung mass alone.

Suspension Force Effects

Movement of a wheel along the line of motion allowed by the suspension kinematics is affected by suspension springs, dampers, bump stops, and anti-sway bars. In each case,

some of the force generated by a component (e.g., a spring) acts to move the wheel, affecting the transfer of mechanical energy to and from the sprung mass. In addition, some of the force is reacted at other points or in other directions that do not move and therefore cannot affect the transfer of mechanical energy. For example, consider the spring shown in the suspension of Figure G.3. If the wheel moves vertically an amount of Δ relative to the body, the spring is compressed by a lesser amount, say, for example, $R_S \Delta$, where R_S is a coefficient that defines the mechanical advantage of the spring relative to the wheel. The spring exerts a force F_S on the lower control arm. Some of the force is reacted at the connection to the body, and some is reacted at the wheel by the vertical tire force, as shown in the figure. Conservation of work requires that the change in force at the wheel center multiplied by its movement must be equal to the change in spring force, multiplied by its change in compression. Thus, the effect of the spring at the wheel is $R_S F_S$. A similar analysis can be made for the damper, using a different ratio R_d.

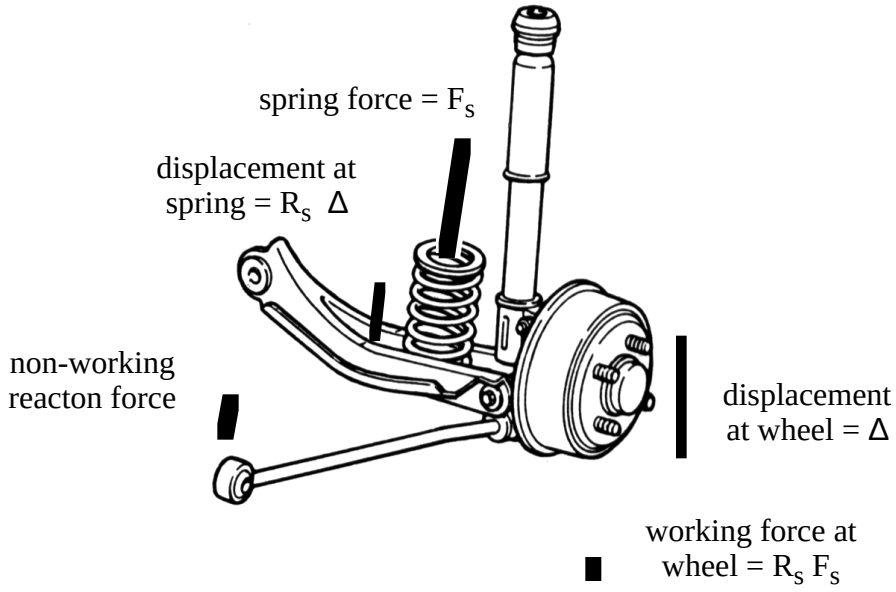

spring force = F_S
displacement at spring = $R_S \Delta$
non-working reacton force
displacement at wheel = Δ
working force at wheel = $R_S F_S$

Figure G.3. Mechanical advantage of suspension component.

This principle of mechanical advantage is used to include components such as springs and dampers in the vehicle dynamics model without requiring details about their points of attachment or the complex suspension linkage geometry. The effect of a suspension component at the wheel is calculated in three steps:

1. multiply the suspension compression (measured at the wheel) by the kinematic ratio to determine the compression at the component,
2. apply a known functional relationship (e.g., spring force vs. compression) to determine the force generated by the component, and
3. multiply the component force by the kinematic ratio to obtain an effective vertical force at the wheel.

For a linear spring, the three steps can be combined to define an effective spring rate at the wheel: $K_S R_S^2$. For nonlinear relations, it is necessary to perform all three steps.

A similar treatment is made for the shock absorber, using the derivative of the suspension displacement, the damper mechanical advantage R_d, and a functional relation between damper force and stroke rate.

The effect of the anti-sway bar is modeled for the independent suspension with a linear spring between the two wheels linked by the bar. The two points are on the two wheels, the direction of the force is $\mathbf{s}_Z$, and the magnitude is a spring rate multiplied by the vertical movement difference between the two points. The parameter is a torsional spring rate. It is converted to a translational spring rate by the simulation solver program.

Summary of Major Model Variables

The CarSimEd models are documented in README text files in the folder with the CarSimEd DLL, MATLAB PIF, and SIMULINK model. (The folders are contained in the CarSimEd `Matlab` folder.)

Input and output variables from the README file are listed in Appendix J. Other information is listed below. The details are presented for the CMEX version of the all-independent vehicle model. (The only differences between the CMEX and stand-alone models are that the CMEX versions include additional inputs that can be defined in the SIMULINK work space.)

Multibody Model Descriptions

CarSimEd 3D vehicle is represented mathematically by 33 ordinary differential equations that describe its kinematical and dynamical behavior. It is composed of 9 bodies, has 10 multibody degrees of freedom, 10 multibody coordinates, 9 auxiliary coordinates, 10 multibody speeds, 4 auxiliary speeds, and has 22 active forces and 4 active moments.

9 Bodies

Entire vehicle (S); parent=N; trans coords = Q(1), Q(2), Q(3); rot coords = Q(4), Q(5), Q(6)

LF wheel (WLF); parent=S; trans coord = Q(7)

LR wheel (WLR); parent=S; trans coord = Q(8)

RF wheel (WRF); parent=S; trans coord = Q(9)

RR wheel (WRR); parent=S; trans coord = Q(10)

WLFN; parent=S

WRFN; parent=S

WLRN; parent=S

WRRN; parent=S

Multibody Coordinates

XO: Abs. X trans. of S0 :Q(1) (m)

YO: Abs. Y trans. of S0 :Q(2) (m)

ZO: Abs. Z trans. of S0 :Q(3) (m)

YAW: Abs. Z rot. of Spp :Q(4) (deg)

PITCH: Y rot. of Sp rel. to Spp :Q(5) (deg)

ROLL: X rot. of S rel. to Sp :Q(6) (deg)

JNC_WLF: Trans. of WLF0 rel. to WLFJ :Q(7) (m)

JNC_WLR: Trans. of WLR0 rel. to WLRJ :Q(8) (m)

JNC_WLR: Trans. of WRF0 rel. to WRFJ :Q(9) (m)

JNC_WRR: Trans. of WRR0 rel. to WRRJ :Q(10) (m)

ROTLF: Rotation angle of LF wheel: Q(11) (deg)

ROTLR: Rotation angle of LR wheel: Q(12) (deg)

ROTRF: Rotation angle of RF wheel: Q(13) (deg)

ROTRR: Rotation angle of RR wheel: Q(14) (deg)

TANLF: Slip angle tan of LF tire, with time lag: Q(15) (-)

TANRF: Slip angle tan of RF tire, with time lag: Q(16) (-)

TANLR: Slip angle tan of LR tire, with time lag: Q(17) (-)

TANRR: Slip angle tan of RR tire, with time lag: Q(18) (-)

IVERR: Integral of velocity error for speed controller: Q(19) (m)

Independent Speeds

VXS: Vehicle forward speed :U(1) (m/s)

VYS: Abs. Y trans. speed of SCMC :U(2) (m/s)

VZS: Abs. Z trans. speed of SCMC :U(3) (m/s)

AVZ: Abs. Z rot. speed of S :U(4) (deg/s)

AVY: Abs. Y rot. speed of S :U(5) (deg/s)

AVX: Abs. X rot. speed of S :U(6) (deg/s)

JNCR_WLF: Trans. speed of WLF0 rel. to S :U(7) (m/s)

JNCR_WLR: Trans. speed of WLR0 rel. to S :U(8) (m/s)

JNCR_WLR: Trans. speed of WRF0 rel. to S :U(9) (m/s)

JNCR_WRR: Trans. speed of WRR0 rel. to S :U(10) (m/s)

AVY_LF: Spin rate of LF wheel: U(11) (rev/s)

AVY_LR: Spin rate of LR wheel: U(12) (rev/s)

AVY_RF: Spin rate of RF wheel: U(13) (rev/s)

AVY_RR: Spin rate of RR wheel: U(14) (rev/s)

22 Forces

FSLF: LF suspension spring force; Acts on the entire vehicle from the lf wheel through LF upper suspension attachment point. Magnitude = FM(1); Direction = [sz].

FSLR: LR suspension spring force; Acts on the entire vehicle from the lr wheel through LR upper suspension attachment point. Magnitude = FM(2); Direction = [sz].

FSRF: RF suspension spring force; Acts on the entire vehicle from the rf wheel through RF upper suspension attachment point. Magnitude = FM(3); Direction = [sz].

FSRR: RR suspension spring force; Acts on the entire vehicle from the rr wheel through RR upper suspension attachment point. Magnitude = FM(4); Direction = [sz].

FAUXF: Front anti-sway-bar force; Acts on the lf wheel from the rf wheel through mass center of the lf wheel. Magnitude = -FM(5); Direction = [sz].

FAUXR: Rear anti-sway-bar force; Acts on the lr wheel from the rr wheel through mass center of the lr wheel. Magnitude = -FM(6); Direction = [sz].

FDLF: LF suspension damper force; Acts on the entire vehicle from the lf wheel through LF upper suspension attachment point. Magnitude = FM(7); Direction = [sz].

FDLR: LR suspension damper force; Acts on the entire vehicle from the lr wheel through LR upper suspension attachment point. Magnitude = -FM(8); Direction = [sz].

FDRF: RF suspension damper force; Acts on the entire vehicle from the rf wheel through RF upper suspension attachment point. Magnitude = FM(9); Direction = [sz].

FDRR: RR suspension damper force; Acts on the entire vehicle from the rr wheel through RR upper suspension attachment point. Magnitude = FM(10); Direction = [sz].

FXLF: LF longitudinal force; Acts on the lf wheel from the inertial reference through CTCLF. Magnitude = FXLF; Direction = (z(189)*[nx] + z(186)*[ny]).

FXLR: LR longitudinal force; Acts on the lr wheel from the inertial reference through CTCLR. Magnitude = FXLR; Direction = (z(202)*[nx] + z(199)*[ny]).

FXRF: RF longitudinal force; Acts on the rf wheel from the inertial reference through CTCRF. Magnitude = FXRF; Direction = (z(215)*[nx] + z(212)*[ny]).

FXRR: RR longitudinal force; Acts on the rr wheel from the inertial reference through CTCRR. Magnitude = FXRR; Direction = (z(228)*[nx] + z(225)*[ny]).

FYLF: LF tire side force; Acts on the lf wheel from the inertial reference through CTCLF. Magnitude = FYLF; Direction = (-z(234)*[nx] -z(231)*[ny]).

FYLR: LR tire side force; Acts on the lr wheel from the inertial reference through CTCLR. Magnitude = FYLR; Direction = (-z(240)*[nx] -z(237)*[ny]).

FYRF: RF tire side force; Acts on the rf wheel from the inertial reference through CTCRF. Magnitude = FYRF; Direction = (-z(246)*[nx] -z(243)*[ny]).

FYRR: RR tire side force; Acts on the rr wheel from the inertial reference through CTCRR. Magnitude = FYRR; Direction = (-z(252)*[nx] -z(249)*[ny]).

FZLF: LF vertical tire force; Acts on the lf wheel from the inertial reference through CTCLF. Magnitude = FM(11); Direction = [nz].

FZLR: LR vertical tire force; Acts on the lr wheel from the inertial reference through CTCLR. Magnitude = FM(12); Direction = [nz].

FZRF: RF vertical tire force; Acts on the rf wheel from the inertial reference through CTCRF. Magnitude = FM(13); Direction = [nz].

FZRR: RR vertical tire force; Acts on the rr wheel from the inertial reference through CTCRR. Magnitude = FM(14); Direction = [nz].

4 Moments

MZLF: LF tire aligning moment; Acts on the lf wheel from the inertial reference. Magnitude = MZLF; Direction = [nz].

MZLR: LR tire aligning moment; Acts on the lr wheel from the inertial reference. Magnitude = MZLR; Direction = [nz].

MZRF: RF tire aligning moment; Acts on the rf wheel from the inertial reference. Magnitude = MZRF; Direction = [nz].

MZRR: RR tire aligning moment; Acts on the rr wheel from the inertial reference. Magnitude = MZRR; Direction = [nz].

References

1. M.W. Sayers and D.S. Han. "A Generic Multibody Vehicle Model for Simulating Handling and Braking," *Vehicle System Dynamics*, Vol. 25 supplement, 1996.
2. Segel, L., "Keynote Address: Some Reflections on Early Efforts to Investigate the Directional Stability and Control of the Motor Car," *Transportation Systems*, ASME Publication AMD, Vol. 108, 1990.
3. Segel, L., "Theoretical Prediction and Experimental Substantiation of the Response of the Automobile to Steering Control," The Institute of Mechanical Engineers, London, 1956.
4. Jindra, F. "Mathematical Model of Four-Wheeled Vehicle for Hybrid Computer Vehicle Handling Program," National Highway Traffic Safety Administration, DOT HS 801800, 1976
5. Wade Allen, Henry T. Szostak et. al., Vehicle Dynamic Stability and Rollover, National Highway Traffic Safety Administration, DOT HS 807 956, June 1992.
6. Orlandea, N. and M.A. Chace, "Simulation Of A Vehicle Suspension With The Adams Computer Program," SAE Paper No. 770053, 1977.
7. Antoun, R. J. et. al., "Vehicle Dynamic Handling Computer Simulation – Model Development, Correlation, and Application Using ADAMS," SAE Paper No. 860574, 1986
8. Kortüm,W. Sharp, R.S., eds., *Multibody Computer Codes in Vehicle System Dynamics*, Lisse, Swetz and Zeitlinger, 1993.
9. Sayers, M.W., "Symbolic Vector/Dyadic Multibody Formalism for Tree-Topology Systems." *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 6, Nov/Dec 1991, 1240-1250.
10. Gillespie, T. D., *Fundamentals of Vehicle Dynamics*, SAE, Warrendale, PA, 1992

Appendix H — The Tire Model

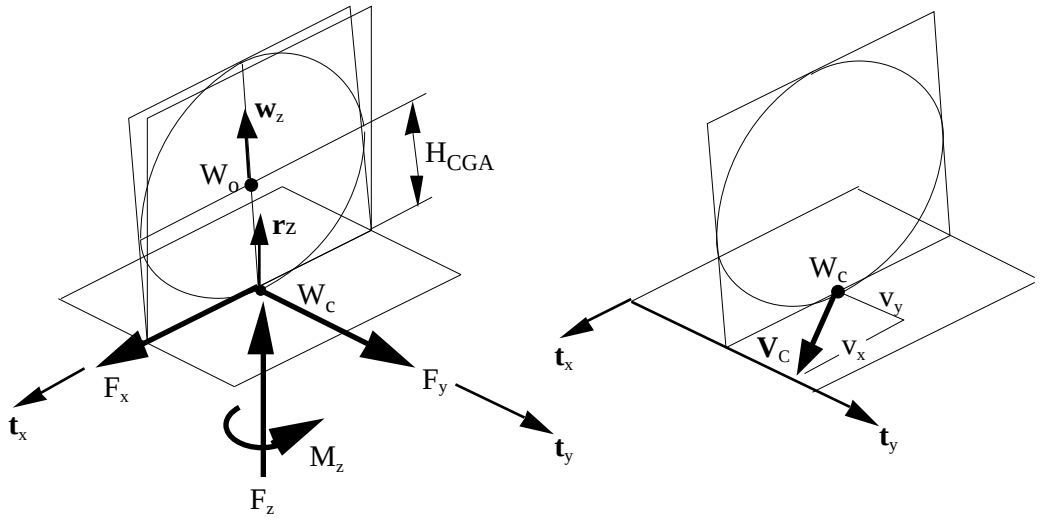


Figure H.1. Tire points and axes.

The interaction of each tire with the ground is modeled by a longitudinal force (F_X), a lateral force (F_Y), a vertical force (F_Z), and an aligning moment (M_Z). F_X, F_Y, F_Z and M_Z are applied to the axles and reacted by the ground, as shown in Figure H.1. Overturning moment (M_X) and rolling resistance moment (M_Y) have an insignificant influence on vehicle braking and handling behavior and are ignored. F_Z is determined by assuming the tire behaves as a linear spring when deformed vertically. F_X, F_Y and M_Z are more complicated to compute.

Tire/Wheel Kinematics

Most of the arithmetic operations in the expressions for tire forces and moments arise from deriving expressions for various kinematical quantities associated with the tires/wheels. It is necessary to (1) define a point where the tire forces act on the multibody model, (2) establish the unit vector directions for the tire X and Y directions ($\mathbf{t}_X$ and $\mathbf{t}_Y$, respectively) given the ground normal ($\mathbf{r}_Z$), and (3) determine expressions for κ and α. These steps are explained below.

Center of Tire Contact

The tire model in CarSimEd makes use of a point called the center of tire contact (CTC) for the definition of the location of the tire on the ground, the calculation of slip angles, and the point of application of tire forces. In the nominal configuration, in which the

vehicle is at rest in equilibrium on a flat, level surface, CTC corresponds to the point W_C shown in Figure H.1.

CTC remains in the ground plane as the wheel moves up and down. For a flat surface, the kinematical expressions for calculating the position and velocity of CTC are well defined, but complex. In nearly all vehicle dynamics models that have been developed by hand (without multibody programs), approximations to CTC for a flat surface have been used. The exact expressions can be obtained easily with the aid of a multibody program. However, the complexity of the exact equations requires many mathematical operations per time step during numerical integration. Most of the complexity arises from terms that are negligible.

When the ground surface is permitted to take on an arbitrary shape, calculating the location of CTC becomes even more complex. This is because $\mathbf{r}_Z$ is a function of the global location of CTC, but at the same time, $\mathbf{r}_Z$ determines the instantaneous tire deflection, which in turn affects the location of CTC. (The general solution of the location of CTC on an arbitrarily shaped surface requires an iterative solution at each time step.)

To obtain shorter run times and avoid the complexities associated with the precise definition of CTC, CarSimEd uses a point W_C fixed in the wheel or axle body (depending on whether the wheel is in an independent or solid axle suspension). Point W_C is used to determine the slip angle, but point CTC is used as the point of application for the resulting tire forces.

Wheel Plane

Each tire has an associated wheel plane that is perpendicular to the wheel spin axis, $\mathbf{w}_Y$. For the front axle, the wheels are steered with angles δ_L and δ_R relative to the axle Y axis. The spin axis for a wheel is:

$$\mathbf{w}_Y = \cos(\delta)\,\mathbf{a}_Y - \sin(\delta)\,\mathbf{a}_X \tag{1}$$

Point W_C is located in the wheel plane and is nominally coincident with the precise definition of CTC (see Figure H.1). The X, Y and Z coordinates of W_C are zero for independent suspensions. For the solid-axle suspension, the coordinates in the axle body are:

$$\begin{aligned} W_{CX} &= 0 \\ W_{CY} &= \pm \frac{L_{TK}}{2} \\ W_{CZ} &= -H_{CGA} \end{aligned} \tag{2}$$

Here, L_{TK} is the axle track to the center plane of the wheel. The algebraic signs of the terms in the Y coordinate depend on the side of the axle (left: $+\frac{L_{TK}}{2}$, right: $-\frac{L_{TK}}{2}$).

Ground Geometry

The coordinates of the absolute position of W_C in the global frame N are:

$$\begin{aligned} X_C &= \mathbf{n}_X \cdot \mathtt{pos}(W_C) \\ Y_C &= \mathbf{n}_Y \cdot \mathtt{pos}(W_C) \\ Z_C &= \mathbf{n}_Z \cdot \mathtt{pos}(W_C) \end{aligned} \tag{3}$$

For a three-dimensional ground surface, the ground normal $\mathbf{r}_Z$ is determined as a function of the location of point $W_{C:}$

$$\mathbf{r}_Z = X_{RZ}\,\mathbf{n}_X + Y_{RZ}\,\mathbf{n}_Y + Z_{RZ}\,\mathbf{n}_Z \tag{4}$$

where

$$[Z_{GROUND}, X_{RZ}, Y_{RZ}, Z_{RZ}] = \text{func}(X_C, Y_C) \tag{5}$$

and func(X_C, Y_C) is a function of X_C and Y_C that returns the four variables shown in braces. For a flat level surface, $\mathbf{r}_Z = \mathbf{n}_Z$.

Vertical Tire Deflection

Vertical tire deflection (Δ_T) is needed to calculate F_Z. Δ_T is the vertical distance between point W_C and the road, in the plane of the wheel:

$$\Delta_T = Z_{GROUND} - Z_C \tag{6}$$

F_X, F_Y, and F_Z act on the axle or wheel through a point CTC that is nominally coincident with W_C, but which moves vertically by the displacement Δ_T in the $\mathbf{a}_Z$ direction. This movement of the force application point provides a slight correction to the moment arm of the forces.

Tire Axes

The tire X and Y axes, $\mathbf{t}_X$ and $\mathbf{t}_Y$, lie in the plane of the road ($\mathbf{t}_Z = \mathbf{r}_Z$). $\mathbf{t}_X$ is defined as being perpendicular to the wheel spin axis. Thus,

$$\mathbf{t}_X = \frac{\mathbf{w}_Y \times \mathbf{r}_Z}{|\mathbf{w}_Y \times \mathbf{r}_Z|} \qquad \mathbf{t}_Y = \frac{\mathbf{r}_Z \times \mathbf{t}_X}{|\mathbf{r}_Z \times \mathbf{t}_X|} \tag{7}$$

Wheel Spin

The wheel spin equation is a simple balance of moments about the spin axis. Three moments are considered: drive torque M_{YDV}, brake torque M_{YBK}, and the moment of longitudinal tire force acting with a moment arm equal to the deflected tire radius.

The wheel spin, ω, is described with a differential equation:

$$\dot{\omega} = \frac{M_{YDV} - \left[F_X (H_{CGA} - \Delta_T) + \text{sign}(M_{YBK}, \omega)\right]}{I_W} \quad (8)$$

where $H_{CGA} - \Delta_T$ is the instantaneous tire radius (moment arm of F_X about the wheel center), and I_W is the polar moment of inertia of the spinning wheel.

The wheel spin is integrated to calculate the wheel rotation angle needed for making animations of the rotating wheels.

Longitudinal Slip

Longitudinal slip (κ) is normally defined as

$$\kappa = \frac{\omega}{\omega_o} \quad (9)$$

where ω_o is the zero-slip angular speed of the wheel:

$$\omega_o = \left|\frac{V_x}{H_{CGA}}\right| \quad (10)$$

The slip angle (α) for each tire is defined in terms of the X and Y velocity of W_C, V_X and V_Y, expressed in the ground plane:

$$v_X = \mathbf{t}_X \bullet \texttt{vel}(W_C) \qquad v_Y = \mathbf{t}_Y \bullet \texttt{vel}(W_C) \quad (11)$$

The slip angle is the arc-tangent of the ratio: v_Y / v_X.

Tire Relaxation

Tires develop shear forces in response to deformation of the tire structure. The forces do not develop instantaneously, but build as the tire rolls [1]. For example, Figure H.2 shows how F_Y builds in response to a step change in slip angle.

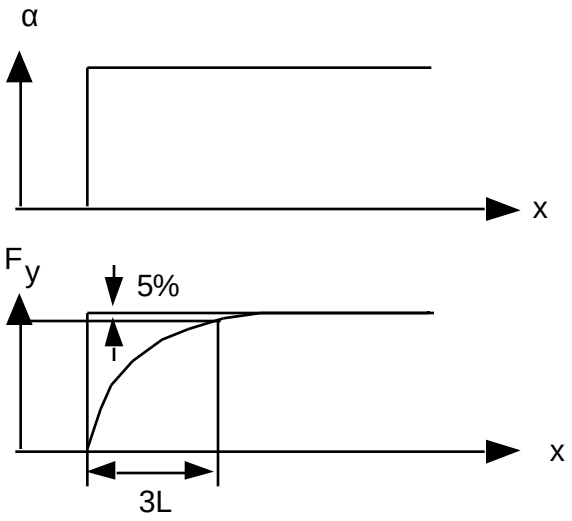


Figure H.2. Tire relaxation.

Two methods are commonly used for including the tire lag in a vehicle model: (1) use a tire model with the dynamics built in, or (2) use a static (steady-state) tire model with a separate filter to account for the lag. The second approach is used in CarSimEd because it offers two practical advantages. First, it allows the use of any static tire model from the literature independently of the method used to introduce lag. Second, it simplifies the calculation of the kinematical variables used as inputs to the tire model. Lag is introduced into the slip angle such that the instantaneous response calculated for the lagged slip angle yields the lagged side force and aligning moment.

A method described by Bernard [2] is used to account for the lag in tire response. In this method, a lagged slip angle, α_L is defined as the arc tangent of an auxiliary state variable, τ.

$$\alpha_L = \tan^{-1}(\tau) \tag{12}$$

A state variable is added for each wheel and defined with a first-order differential equation:

$$\frac{d\tau}{dt} = \frac{|V_X|}{L_{RELAX}}\left(\frac{V_Y}{|V_X|} - \tau\right)$$

$$= \frac{V_Y - \tau|V_X|}{L_{RELAX}} \tag{13}$$

The absolute value of V_x is used to maintain continuity in case the vehicle spins out and V_x assumes a negative value

At very low speed, the above equation does not work well because the time lag goes to infinity. Therefore the “gain” on the derivative is increased according to the equation:

$$\frac{d\tau}{dt} \leftarrow \frac{d\tau}{dt} \cdot \left(1 + 50 \cdot \left(1 - \frac{|v_x|}{v_{\alpha o}}\right)\right) \tag{14}$$

where $v_{\alpha o}$ is a cut-off speed—a parameter that can be set by the user.

Tire Forces and Moments

Once expressions are obtained for the needed kinematical quantities, the tire force and moment magnitudes can be calculated. F_Z must be calculated first, because F_X, F_Y and M_Z depend upon it.

Vertical Tire Force

Vertical tire force is proportional to the tire deflection:

$$F_Z = \max\left(0,\ F_{ZSTAT} + K_T \Delta_T\right) \tag{15}$$

where F_{ZSTAT} is the static tire load and K_T is the vertical tire stiffness.

The above definition of F_Z establishes the force to be in equilibrium when the vehicle is in the nominal configuration. The use of the `max` function prevents the magnitude of the force from going negative when the tire leaves the ground.

Pure Longitudinal and Lateral Slip

A relatively simple tire model is used in CarSimEd to predict the linear cornering behavior and the nonlinear limit behavior. F_{xo}, F_{yo}, and M_{zo} are values of longitudinal force, lateral force, and aligning moment that would be obtained in the absense of friction limits. They are obtained first, and then the values are reduced to account for frictional limits using a normalization method described by Radt [3] and summarized in the equations below.

Longitudinal force is mainly determined by the brake torque and rolling radius for sub-limit behavior. The tire seeks a spin in which the torque balance of equation (11) is maintained. The tire model mainly determines how much longitudinal slip occurs to generate the longitudinal force. For the purpose of producing a longitudinal slip that is within an order of magnitude of the correct one, a simple equation for longitudinal tire force is used:

$$F_{xo} = K_{fx}\,\kappa \tag{16}$$

Lateral force is determined for a the combined effect of camber and lateral slip:

$$F_{yo} = -K_{fy}(F_Z)\,\alpha + K_{f\gamma}\,\gamma \tag{17}$$

where K_{fy} is a nonlinear table-lookup that defines cornering stiffness as a function of load. Aligning moment is defined as the product of lateral force and a moment arm called pneumatic trail:

$$M_{zo} = F_{yo} \cdot L_{trail}(F_Z) \tag{18}$$

where L_{trail} is a nonlinear table-lookup that defines trail as a function of load.

To account for friction limits, the following terms are defined. First, lateral slip and inclination angle are combined:

$$T_{eff} = \tan(\alpha - \frac{K_{Fy\gamma}}{K_{Fy}}\gamma) \tag{19}$$

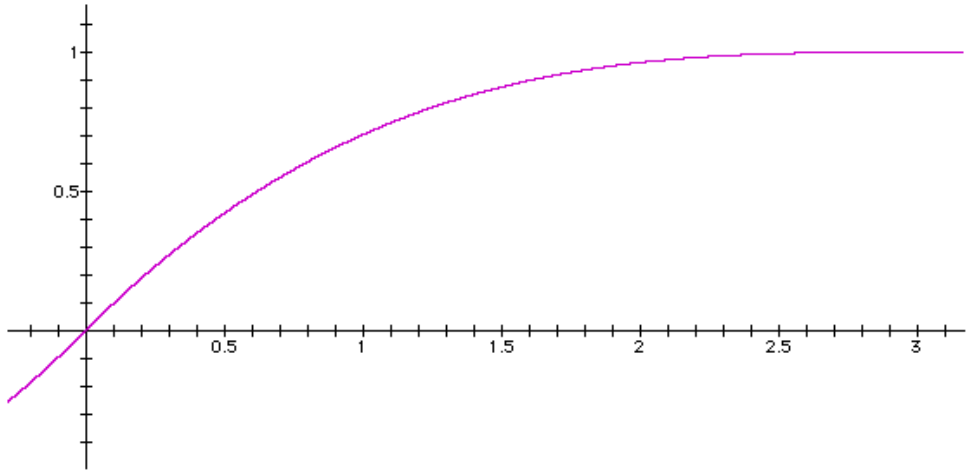
Next, longitudinal and lateral slip are combined:

$$K = \frac{\sqrt{\left(K_{Fy}T_{eff}\right)^2 + \left(K_{Fx}\kappa\right)^2}}{\mu F_Z} \tag{20}$$

where μ is the tire/road friction coefficient.

A normalized force is defined to provide a shape for the transition from linear to sliding.

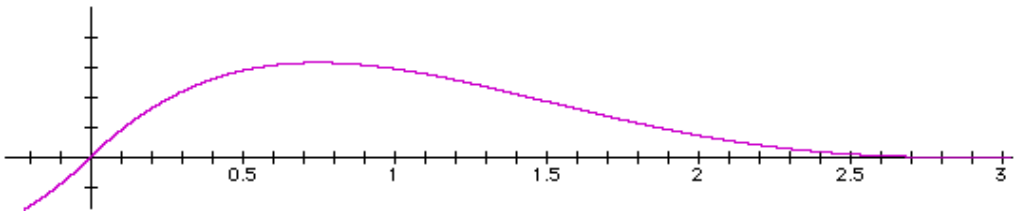
$$\overline{F} = K - \frac{|K|K}{3} + \frac{K^3}{27} \tag{21}$$



Plot of equation 20.

A different shape function is used to define normalized aligning moment.

$$\overline{M} = K - |K|K + \frac{K^3}{3} - \frac{|K|K^3}{27} \tag{22}$$



Plot of equation 21.

A coefficient "ν" is defined to transition between linear and full sliding:

1. If (K > 2π) then

$$\nu = 1$$

1. else

$$\nu = \frac{1}{2}\left(1 + \frac{K_{Fy}}{K_{Fx}} - \left(1 - \frac{K_{Fy}}{K_{Fx}}\right)\cos\left(\frac{K}{2}\right)\right) \tag{23}$$

A force scaling factor is defined:

$$F_0 = \frac{\mu F_Z}{\sqrt{\kappa^2 + \left(\nu T_{eff}\right)^2}} \tag{24}$$

The scaled forces and moment are:

$$\begin{aligned} F_x &= \kappa \quad \bar{F} \quad F_0 \\ F_y &= -\nu \quad T_{eff} \quad \bar{F} \quad F_0 \\ M_Z &= \nu \quad L_{trail} \quad T_{eff} \quad \bar{M} \quad F_0 \end{aligned} \tag{25}$$

Camber Effect

The small effect of inclination is handled by modifying the slip angle, by an amount:

$$\alpha_{eff} = \tan^{-1}(\tau) + \gamma \bullet R_\gamma \qquad\qquad R_\gamma = \left|\frac{K_\gamma}{K_\alpha}\right| \tag{26}$$

where K_γ is a linear camber-thrust coefficient and K_α is the cornering stiffness. Both coeffients are evaluated at the instant vertical load F_z.

Low-Speed Exceptions

Models of the rolling tire can become unrealistic when the rolling stops. One problem is numerical—equations might have the speed in the denominator of an expression. For example, the longitudinal slip definition (see Equation 9) is singular at zero speed. This problem is solved by using limiting the speed used in the equation to a small ε value:

$$\text{If } \frac{V_x}{H_{CGA}} < \varepsilon \text{ Then } \omega_O = \text{sign}(\varepsilon, V_X) \tag{27}$$

Low-Speed Instabilities

A general challenge in mechanical simulation occurs whenever a large force or moment is largely insensitive to the magnitude of motion speed, but acts in the direction opposite

the motion. For example, braking torque resists the direction of wheel spin but is not dependent on the magnitude of the spin rate. As long as the wheel is spinning, the brake torque causes angular acceleration that slows the wheel. Each time step, the numerical integration calculations predict a negative change in speed. But when the wheel spin reaches zero, the calculated increment in speed is likely to overshoot zero, causing the predicted wheel spin to be negative for the next time step. Then, the brake torque acts in opposition to the negative speed and causes a positive acceleration, leading to a positive change in speed, and a positive speed for the next time step. Each time step, the wheel spin and braking moment change sign. Because the amount of brake torque does not diminish with the spin magnitude, the oscillation does not diminish either.

The above problem occurs when friction is modeled. It is sometimes called a “bang-bang” instability because the large friction force is applied fully in opposite directions with each time step.

In the real mechanical system, brake torque for a locked up wheel is a reaction torque, exactly equal to the amount necessary to prevent motion. There is no oscillation.

There are two friction-like instabilities associated with zero speed.

First, the instability described above exists when the wheel spin approaches zero due to brake torque. An alternate to Equation 8 is used when the brake torque is greater than the moment of the tire braking force plus the driving moment, because it is inevitable that the wheel is about to lock up. Specifically,

If $M_{YBK} > |F_X \bullet (H_{CGA} - \Delta_T)|$ And $M_{YBK} > |M_{YDV}|$ And $|\omega| < \omega_{ao}$ Then

$$\dot{\omega} \leftarrow \dot{\omega} \cdot \frac{|\omega|}{\omega_{ao}} \tag{28}$$

where ω_{ao} is a low-speed threshold for spin acceleration.

> Note: The low-speed threshold is provided by the user with the parameter `VLOW_SPINA` with units of kph. The undeflected rolling radius is used to convert `VLOW_SPINA` to ω_{ao}.

Another instability occurs when a wheel is locked and the vehicle speed approaches zero. The longitudinal slip (see Equation 9) predicts full longitudinal force in opposition to the vehicle longitudinal motion. To prevent violent longitudinal oscillations when the vehicle should be at rest, the longitudinal slip is reduced, to generate reduced longitudinal force and to “use up” less of the combined lateral and longitudinal slip. In order for the alternate equation to be applied, the wheel must be moving slowly and locked up (or approaching lockup).

If $|\omega| < |\omega_o|$ And $|\omega_o| < \omega_{KO}$ Then

$$\kappa \leftarrow \kappa \bullet \frac{1}{2}\left[1 - \cos\left(\frac{\omega_o \bullet \pi}{\omega_{KO}}\right)\right] \tag{29}$$

where ω_{KO} is a low-speed threshold.

> Note: The low-speed threshold is provided by the user with the parameter VLOW_KAPPA with units of kph. The nominal rolling radius is used to convert VLOW_KAPPA to ω_{KO}.

Sequence Of Calculations

The vehicle simulation is run by numerically integrating a set of ordinary differential equations. At the beginning of each time step, the values of all of the state variables are known, including the τ variables. The various quantities necessary for the tire equations are calculated in the following sequence:

1. Compute F_Z, which depend only on the state variables.
2. Compute F_X, F_Y and M_Z with the static tire model (table look-up/combined slip theory) using τ.
3. Compute δ, including the effect of steer compliances coupled with tire actions of F_X, F_Y and M_Z.
4. Determine $\mathbf{t}_x$ and $\mathbf{t}_y$.
5. Calculate terms that depend on $\mathbf{t}_x$ and $\mathbf{t}_y$:
   - Apply forces F_X and F_Y to the axles.
   - Calculate the time derivatives of τ.

Sign Conventions

For the above definitions of longitudinal and lateral slip,

1. positive κ generates positive F_X,
2. positive α generates negative F_Y, and
3. positive (small) α generates positive M_Z.

When the longitudinal speed of the wheel center is negative, the conventions for tire X and Y axes would be reversed according to standard tire kinematics conventions. However, in the CarSimEd models, the X axis remains pointing towards the front of the vehicle and the Y axis continues pointing to the vehicle left. Therefore the signs of Fx and Mz are reversed when the wheel speed is negative. The sign of Fy is not reversed due to the use of $|v_X|$ in the definition of tan(α).

References

1. Loeb, J.S. et. al., “Lateral Stiffness, Cornering Stiffness and Relaxation Length of the Pneumatic Tire,” SAE Paper No. 900129, 1990.

2. Bernard, J. E., Clover, C. L., “Tire Modeling for Low–speed and High–speed Calculations,” SAE Paper No. 950311, 1995

3. H. Radt, “Tire Data Treatment,” Chapter 14 of *Race Car Vehicle Dynamics*, W.F. Milliken and D.L. Milliken, SAE, 1995. pp. 473-487.

Appendix I — The CarSimEd Steer Controller

This appendix describes the theory and application of an algorithm used to control a road vehicle to follow a prescribed path. The technique was developed by Charles MacAdam of UMTRI in 1980 [1, 2] and has been used in various forms in many computer programs since then. In recent research for the Federal Highway Administration (FHWA), the algorithm has been streamlined and re-formulated to work with roadway centerline geometry.

Optimal Control Theory

The algorithm is intended to provide optimal control for a linear system:

$$\dot{\mathbf{x}} = A\,\mathbf{x} + B\,u \tag{1}$$

$$y = C\,\mathbf{x} + D\,u \tag{2}$$

where $\mathbf{x}$ is an array of *n* state variables, u is a control input, y is an output variable of interest, and A, B, C, and D are matrices with constant coefficients. The control objective is to determine the optimal value of u to causes the predicted output y(t) to match a target trajectory $y_{target}(t)$, over some previewed time T.

In the above equations, A is an $n \times n$ matrix. In the general case, u and y could be arrays involving more than one control and/or output variable. However, in this derivation, the only case considered is the one in which u and y are scalars. Thus, B is a $n \times 1$ matrix and C is a $1 \times n$ matrix. There is a further simplification, namely, that the output y does not depend explicitly on u. Thus, the D matrix is not used.

If the system has initial conditions $\mathbf{x}_o$ at time t=0 and a constant control input u, its time response has the known solution:

$$\mathbf{x}(t) = e^{At}\,\mathbf{x}_o + \int_0^t e^{A\eta}\,B\,u\,d\eta \tag{3}$$

The term e^{At} is an $n \times n$ matrix called the state transition matrix. Each coefficient in the matrix is the portion of state variable i at time t that is linearly related to state variable j at time 0. It can be calculated with a power series, with eigenvectors, or by numerical integration. (As will be described later, numerical integration is the method used in the driver model.) The product of the state transition matrix (e^{At}) and the array of initial conditions ($\mathbf{x}_o$) is an array of length *n* with the part of each state variable at time t due to the initial conditions of the system at t=0. This is called the free response. The integral defines a contribution to each state variable due to the constant control u over the time interval. It is called the forced response.

Combining Equations 2 and 3 gives the response of the output variable:

$$y(t) = C\,\mathbf{x} = C\,e^{At}\,\mathbf{x}_o + C\,[\int_0^t e^{A\eta}\,d\eta]\,B\,u \tag{4}$$

A free-response array F (a 1x*n* matrix) is defined to simplify some of the following notation. F relates the state variables at time 0 to the resulting output variable y at time t.

$$F(t) = C\,e^{At} \qquad f_i(t) \equiv \frac{\partial y(t)}{\partial x_i(0)} \tag{5}$$

A control-response scalar g is also defined, to relate the control input u over the interval t to the output variable y at time t.

$$g(t) = C\,[\int_0^t e^{A\eta}\,d\eta]\,B \qquad g(t) \equiv \frac{\partial y(t)}{\partial u} \tag{6}$$

Note that g(t) is related to F(t) by integration:

$$\begin{aligned} g(t) &= [\int_0^t C\,e^{A\eta}\,d\eta]\,B \\ &= [\int_0^t F(\eta)\,d\eta]\,B \end{aligned} \tag{7}$$

The response equation, re-written using the newly introduced terms, is:

$$y(t) = F(t)\,\mathbf{x}_o + g(t)\,u \tag{8}$$

To determine the optimal control, a quadratic performance index J is defined,

$$J = \frac{1}{T}\int_0^T \left\{y_{target}(t) - y(t)\right\}^2 W(t)dt \tag{9}$$

where W(t) is an arbitrary weighting function. A control u is considered optimal if it minimizes J—the squared deviation of response variable y(t) relative to the target function $y_{target}(t)$. Because J is quadratic, the minimum occurs when the derivative $\partial J/\partial u$ is zero. The value of u which minimizes J can be found by substituting Equation 8 into 9 and taking the partial derivative of J with respect to u.

$$J = \frac{1}{T}\int_0^T \left\{F(t)\mathbf{x}_o + g(t)u - y_{target}(t)\right\}^2 W(t)dt \tag{10}$$

$$\frac{\partial J}{\partial u} = 0 = \frac{2}{T}\int_0^T \left\{F(t)\mathbf{x}_o + g(t)u - y_{target}(t)\right\} g(t)W(t)dt \tag{11}$$

Solving for u gives the following:

$$u = \frac{\int_0^T \left\{y_{target}(t) - F(t)\,\mathbf{x}_o\right\} g(t)W(t)dt}{\int_0^T g(t)^2 W(t)dt} \tag{12}$$

In practice, the integrals over T can be replaced with finite summations:

$$u = \frac{\sum_{i=1}^{m} \left\{ y_{target_i} - F_i \mathbf{x}_o \right\} g_i W_i}{\sum_{i=1}^{m} g_i^2 W_i} \tag{13}$$

where the time dependencies of Equation 12 are replaced with an index i. The meaning here is that index i applied to F, g, W, and y_{target} refers to the value at time t=iT/m, where m is an integer number of intervals within the preview time T.

A further simplification will be made in the application below; the arbitrary weighting function will be set to unity. The simplified equation is shown below for future reference.

$$u = \frac{\sum_{i=1}^{m} \left\{ y_{target_i} - F_i \mathbf{x}_o \right\} g_i}{\sum_{i=1}^{m} g_i^2} \tag{14}$$

Application to Vehicle Control

The general control method is programmed in the UMTRI driver model to generate a steering wheel angle in a vehicle solver program. The algorithm flow is shown in Figure I.1. Given a target path, described in terms of X and Y coordinates, the driver model computes a steering wheel angle given the current state of the vehicle.

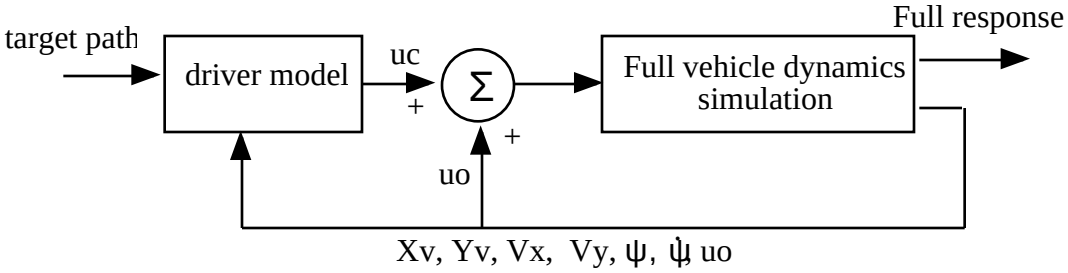


Figure I.1. Algorithm flow.

The vehicle dynamics simulation is usually a complex nonlinear model involving thousands of calculations and tens of degrees of freedom (DOF). It moves on a 3D surface following a prescribed path that may be simple or complex. The start time of the simulation is usually set to zero, and the stop time may be set to anything from a few milliseconds to several minutes.

The complexity of the vehicle simulation is largely hidden from the driver model. In the world of the driver model, the current time is defined as zero, the vehicle is described with a simple model having two degrees of freedom, and the driver control is calculated

to optimize the vehicle response over the preview time T. As shown in the figure, the driver model requires only seven variables as feedback. They are: the X and Y coordinates of the front axle of the vehicle (X_v and Y_v), the vehicle-based longitudinal and lateral components of the velocity vector (V_x and V_y), the yaw angle and yaw rate (ψ and $\dot{\psi}$), and the steer control due to factors outside the driver's direct control (u_o). (The term u_o usually represents steering due to suspension kinematics and compliance.)

The driver control algorithm can be divided into three types of calculations:

1. it synthesizes the target path over the preview time,
2. it calculates the optimal steer u to minimize deviations of the path of a point (the center of the front axle) from a target path and subtracts u_o to obtain the steering needed by the driver, u_c, and
3. it delays the driver steering control u_c by a constant time τ_d, to simulate a well-known dynamic characteristic of human drivers.

Station

The target path is normally provided as a sequence of N_{pts} X and Y coordinates. Station (also called station number) is the distance along a reference line, typically a road centerline. In a road design, station is a spatial independent variable. For any given value of station, there is a unique set of X, Y, and Z coordinates. There may also be other values associated with that position, such as superelevation, road width, etc.

A simplification is made in the driver model that the path is composed of straight lines connecting points defined as a sequence of X-Y values. The X-Y values are specified at run time. The starting value of S is set to match the initial X value:

$$S_1 \equiv X_1 \tag{15}$$

Subsequent values, from i=2 to N_{pts}, are calculated with the recursive equation:

$$S_i = S_{i-1} + [(X_i - X_{i-1})^2 + (Y_i - Y_{i-1})^2]^{1/2} \tag{16}$$

This calculation is done when the X and Y values are read as inputs. After the calculation, each point in the path is represented with three numbers (S, X, Y) instead of two (X, Y). Because the path connecting two points is assumed to be a straight line, the partial derivatives $\partial X/\partial S$ and $\partial Y/\partial S$ can be calculated with finite-difference equations:

$$\left.\frac{\partial X}{\partial S}\right|_i = \frac{\Delta X_i}{\Delta S_i} = \frac{X_{i+1} - X_i}{S_{i+1} - S_i}$$

$$\left.\frac{\partial Y}{\partial S}\right|_i = \frac{\Delta Y_i}{\Delta S_i} = \frac{Y_{i+1} - Y_i}{S_{i+1} - S_i} \tag{17}$$

The forward-difference form of Equation 17 means that the i^{th} derivative applies for the line segment after the i^{th} point (between points i and i+1). Table lookup routines are used to provide X, Y, $\partial X/\partial S$, and $\partial Y/\partial S$ as functions of S at various places in the driver model. For station numbers outside the range of the table, the last values of $\partial X/\partial S$ and $\partial Y/\partial S$ should be held constant to extrapolate X and Y as needed.

Because S is treated as the independent variable for determining the current path geometry, it is essential to know the current value of S. Figure I.2 shows the relationship between the last station number S_o, with corresponding coordinates X_o and Y_o, and the current vehicle location as defined by the coordinates X_v and Y_v.

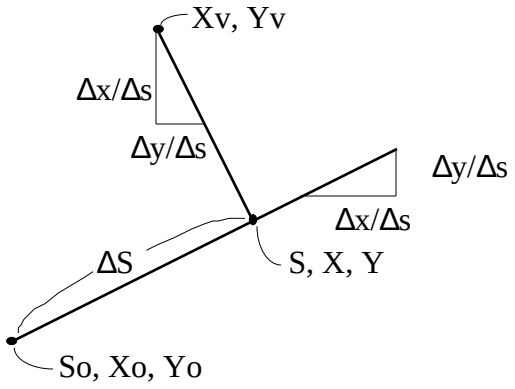


Figure I.2. Geometry used to calculate new station.

The new station S is

$$S = S_o + \Delta S \tag{18}$$

where

$$\Delta S = \frac{(X_V - X_o)\frac{\Delta x}{\Delta s} + (Y_V - Y_o)\frac{\Delta y}{\Delta s}}{\left(\frac{\Delta x}{\Delta s}\right)^2 + \left(\frac{\Delta y}{\Delta s}\right)^2} \tag{19}$$

To calculate the optimal steer control u_C with Equation 14, the target position is needed at each point being considered in the summation. The value of S obtained with Equation 18 is the station number of the vehicle at its current position. The station for a target location is

$$S_{targ,i} = S + \frac{iV_x T}{m} \tag{20}$$

where i = 1,... m, and V_X is the forward vehicle speed, assumed within the world of the driver model to be constant .

Target Position

The controller calculations are made using a special axis system, shown in Figure I.3. At time t=0, the origin of the X and Y axes coincide with the center of the vehicle front axle, and the axes are aligned with the longitudinal and lateral axes of the vehicle. In the world of the driver controller, the vehicle movements will be predicted relative to these axes. The axes are fixed in the inertial reference, and are rotated from the inertial axes by the

vehicle yaw angle, ψ. The initial lateral displacement of the vehicle and the initial yaw angle are both zero in the driver model axis system.

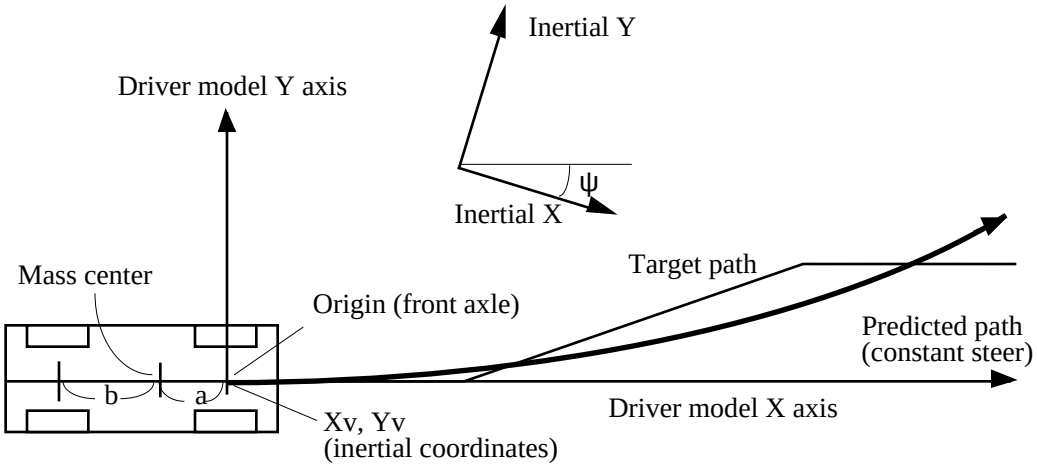


Figure I.3. Axis system of driver model.

The target lateral translation in this coordinate system is calculated by first getting the inertial X and Y coordinates of the path as functions of the station at the target location (S_{targ}), and then applying the transformation

$$Y_{targ} = [Y(S_{targ}) - Y_V]\cos(\psi) - [X(S_{targ}) - X_V]\sin(\psi) \tag{21}$$

Calculating the Optimal Control

Within the world of the driver model, the vehicle always is located at the origin of the axes shown in Figure I.3, the time is 0, and the target path is known from time zero to the preview time, T. The vehicle speed (V_x) is constant and the vehicle is assumed to be described by four state variables:

x_1 = Y coordinate of the vehicle mass center, in the driver model axis system,
x_2 = Yaw angle of vehicle, in the driver model axis system,
x_3 = Vy, the lateral component of velocity in the vehicle axis system, and
x_4 = Yaw rate.

The A and B matrices are defined as follows:

$$A = \begin{bmatrix} 0 & 0 & 1 & V_x \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \dfrac{-(C_f + C_r)}{MV_x} & \dfrac{C_r b - C_f a}{MV_x} - V_x \\ 0 & 0 & \dfrac{C_r b - C_f a}{I_{zz} V_x} & \dfrac{-(C_f a^2 + C_r b^2)}{I_{zz} V_x} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ \dfrac{C_f}{M} \\ \dfrac{C_f a}{I_{zz}} \end{bmatrix} \tag{22}$$

where C_f and C_r are tire cornering stiffness coefficients for the front and rear axles, M is the total vehicle mass, a is the distance from the front axle to the mass center, b is the

distance from the rear axle to the mass center, V_x is the forward component of vehicle speed, and I_{zz} is the polar moment of inertia of the vehicle in yaw. The control u is the steer angle of the front wheels. (Terms can be added for 4-wheel steer.)

The output variable of interest is the lateral position of the front axle. Thus, the C matrix is defined as:

$$C = \begin{bmatrix} 1 & a & 0 & 0 \end{bmatrix} \tag{23}$$

The task performed by the driver model is to calculate a new value of steering angle at the front wheels as the simulation proceeds, using Equation 14 and the target location from Equation 21. Consider the terms needed to make the calculation. The solution involves a summation over m intervals (m is presently programmed as 10). For a given speed, the relation between interval i and time is fixed (t=iT/m). Each time the driver model is called, it is provided $\mathbf{x}_o$ and the information needed to determine y_{targ} for each interval. The coefficients F_i, g_i, and g_i^2 depend only on speed, and do not need to be re-computed unless the vehicle speed has changed.

The free-response coefficients in the array F_i define the lateral position of the vehicle at the end of interval i due to non-zero initial conditions. Although the internal 2-DOF vehicle model has four state variables, the choice of axis systems simplifies the calculations. Figure I.3 shows that the initial values of x_1 (lateral coordinate Y) and x_2 (yaw angle ψ) are identically zero in the axis system of the driver model. Therefore, only two coefficients are needed in array F_i: f_{1i} accounts for initial lateral velocity (x_3), and f_{2i} accounts for initial yaw rate (x_4).

The coefficient f_{1i} represents the value of y at the end of interval i, for an initial value of $x_3 = 1$. The values are calculated for i=1,...,m using numerical integration. The initial values of all state variables except x_3 are set to zero and x_3 is set to unity. The 2-DOF model is then simulated using an Euler integrator from t=0 to t=T, and values of the lateral position are saved at the m locations used in the summation. The process is repeated to determine the values of f_{2i}, except that the initial conditions for the 2-DOF model are that x_4=1 and all other variables are 0.

The free-response coefficients in F_i are also used to compute the control response coefficients g_i, using Equation 7. (The coefficient g_i defines the deviation of the vehicle at the end of interval i due to a sustained steer angle of unity.) An extra state variable is added whose derivative is y. Its integral at the end of each interval i is multiplied by B_3 for the initial condition of $x_3 = 1$, and its integral for the initial x_4=1 is multiplied by B_4. The weighted sum is g_i.

Equation 14 is used to determine the optimal control steer. The steer due to factors other than the driver is provided as u_o, and subtracted from the optimal control to obtain the steer needed by the driver, u_c. The driver steer angle, applied to the front wheels, is divided by a steering gear ratio to obtain the corresponding angle at the steering wheel.

Transport Delay

The steer angle from the driver, u_c, is given a pure transport delay to simulate the neuromuscular delay of a human driver. Previous research has shown the transport delay

to be an important parameter in determining the dynamics of the closed-loop man-machine system [1].

References

1. MacAdam, C.C. "Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 11, June 1981.

2. MacAdam, C.C. "An Optimal Preview Control for Linear Systems," Journal of Dynamic Systems, Measurement, and Control, ASME, Vol. 102, No. 3, Sept. 1980.

Appendix J — Model Input and Output Variables

The main output of the vehicle solver programs in CarSimEd is a pair of files (extensions = ERD and BIN) with time histories of variables computed during the simulation. By convention, the two files together are called an ERD file. (See Appendix C for a full description of the ERD format.)

The simulation is performed by calculating new values of vehicle variables as functions of simulated time. The ERD file contains a large table of numbers, where each column corresponds to a different variable, and each row corresponds to a value of time. The values from this file are read by the plotter and animator programs in CarSimEd.

Types of Output Variables

The ERD file contains variables that fall into three groups:

1. Vehicle variables that characterize its behavior during the run.
2. Additional variables needed to create animations for viewing vehicle motions.
3. Controls and disturbances that are the inputs to the mathematical vehicle model.

Although the ERD file is organized by discrete time steps, all variables can be plotted against any other variable.

Output variables have short and long names, plus other information that is used by the plotter. However, they are identified by their short names, which are defined in the ERD file standard as having exactly 8 characters. (Shorter names are padded with blanks.)

The variables are named using a convention that is summarized in Table J.1. Names can have up to three parts:

1. A prefix of first one or more letters to identify the type of variable (Force, Acceleration, etc.).
2. A following letter further clarifies the type of variable. For example the letters, *x, y,* and *z* are used to identify directions. The letters *s* and *d* are used to identify Spring and Damper forces.

> Notes: Many quantities of interest are vectors. These include acceleration, angular velocity, and tire force. The ERD file contains components of these vectors, with the component being identified with an x, y, or z after the prefix.

> The letters c and t are used for centripital and tangental when used to identify acceleration components.

3. The remaining characters identify the part of the vehicle with which the quantity is associated. The most common are LF, LR, RF, and RR to identify the four wheels (Left-Front, Left-Rear, Right-Front, and Right-Rear, respectively).

Table J.1. Naming convention for output variables.

Prefix	2nd Part	Example	Description
AA	x, y, z	`AAy_LF`	Angular acceleration component
Alpha		`Alpha_LF`	Tire lateral slip angle
AV	x, y, z	`Avx`	Angular velocity component
A	x, y, z	`Ax`	Acceleration component
Beta		`Beta`	Vehicle lateral slip angle
F	d,s,x,y,z	`Fd_LF`	Force component.
Jounce		`JounceLF`	Suspension compression
Kappa		`Kappa_LF`	Tire longitudinal slip
M	x, y, z	`Mybrk_LF`	Moment component (torque)
Mu		`Mu_LF`	Tire-road friction coefficient
P	brk	`Prk_con`	Pressure
Pitch		`Pitch`	Pitch angle of rigid body
Roll		`Roll_WLF`	Roll angle of rigid body
Rot		`Rot_WLF`	Rotation angle of wheel about Y axis (similar to pitch, except it involve very large angles)
Steer		`Steer_LF`	Steer angle
V	x, y, z	`Vx_LF`	Velocity component of a point, or, for wheels, spin converted to velocity
X, Y, Z		`X_WLF`	Coordinate of a point
Yaw		`Yaw_WLF`	Yaw angle of rigid body

Animator Variables

In order to locate and orient a rigid body in 3D space, six independent variables are needed: X-Y-Z coordinates and three rotation angles. For example, the car model has five independent moving parts (the body and four wheels). Each requires six variables to animate. Therefore 30 variables in the ERD file are needed to produce full animations.

As shown in Table J.1, the coordinate variables begin with the letters `X`, `Y`, and `Z`. The rotation variables begin with the names `Yaw`, `Pitch`, `Roll`, and `Rot`. `Pitch` and `Rot` both involve the same conceptual rotation (about a Y axis), however the `Rot` variables apply to the rolling wheels and have units of revolution, rather than degrees.

Inputs

All vehicle controls that involve the driver are available for plotting: the steering wheel angle (`Steer_SW`), the throttle (`Throttle`), and the brake input (`Pbrk_con`).

Inputs and Output Variables

The output variables written into the ERD file are also defined as outputs of the CarSimEd S-Function for use in SIMULINK. In order to view the variables from within SIMULINK, it is necessary to know the index number. This information is summarized in a README text file in each folder in the CarSimEd `Matlab` folder.

Along with the block of output variables, the SIMULINK versions of the CarSimEd model include a block of input variables. These are also listed in the README file. The following two subsections list the inputs and outputs for the 3D car model. The text is taken straight from the file `Matlab\cmx_mdl\Readme.txt`.

7 Input Variables

INPT(1): INPT_STEER, (deg), External steering wheel input

INPT(2): INPT_RSTEER, (deg), External rear wheel steer angle input

INPT(3): INPT_TBRK_LF, (N-m), External brake torque input (LF wheel)

INPT(4): INPT_TBRK_RF, (N-m), External brake torque input (RF wheel)

INPT(5): INPT_TBRK_LR, (N-m), External brake torque input (LR wheel)

INPT(6): INPT_TBRK_RR, (N-m), External brake torque input (RR wheel)

INPT(7): INPT_THROTTLE, (-), External throttle input

195 Output Variables

OUT(1) "AAy_LF" (rev/s2), LF wheel angular acceleration, gen-name = Wheel angular acceleration, rigibody name = LF wheel.

OUT(2) "AAy_LR" (rev/s2), LR wheel angular acceleration, gen-name = Wheel angular acceleration, rigibody name = LR wheel.

OUT(3) "AAy_RF" (rev/s2), RF wheel angular acceleration, gen-name = Wheel angular acceleration, rigibody name = RF wheel.

OUT(4) "AAy_RR" (rev/s2), RR wheel angular acceleration, gen-name = Wheel angular acceleration, rigibody name = RR wheel.

OUT(5) "Alpha_LF" (deg), LF wheel slip angle, without lag, gen-name = Slip angle, rigibody name = LF wheel.

OUT(6) "Alpha_LR" (deg), LR wheel slip angle, without lag, gen-name = Slip angle, rigibody name = LR wheel.

OUT(7) "Alpha_RF" (deg), RF wheel slip angle, without lag, gen-name = Slip angle, rigibody name = RF wheel.

OUT(8) "Alpha_RR" (deg), RR wheel slip angle, without lag, gen-name = Slip angle, rigibody name = RR wheel.

OUT(9) "AVx" (deg/s), Body roll rate, gen-name = Roll rate, rigibody name = Body.

OUT(10) "AVy" (deg/s), Body pitch rate, gen-name = Pitch rate, rigibody name = Body.

OUT(11) "AVy_LF" (rev/s), LF wheel angular velocity, gen-name = Wheel angular velocity, rigibody name = LF wheel.

OUT(12) "AVy_LR" (rev/s), LR wheel angular velocity, gen-name = Wheel angular velocity, rigibody name = LR wheel.

OUT(13) "AVy_RF" (rev/s), RF wheel angular velocity, gen-name = Wheel angular velocity, rigibody name = RF wheel.

OUT(14) "AVy_RR" (rev/s), RR wheel angular velocity, gen-name = Wheel angular velocity, rigibody name = RR wheel.

OUT(15) "AVz" (deg/s), Vehicle yaw rate, gen-name = Yaw rate, rigibody name = Vehicle.

OUT(16) "Ax" (g's), Vehicle long. acceleration, gen-name = Longitudinal acceleration, rigibody name = Vehicle CG.

OUT(17) "Ay" (g's), Vehicle lateral acceleration, gen-name = Lateral acceleration, rigibody name = Vehicle CG.

OUT(18) "Az" (g's), Vehicle vertical acceleration, gen-name = Vertical acceleration, rigibody name = Vehicle CG.

OUT(19) "Beta" (deg), Vehicle slip angle, gen-name = Slip angle, rigibody name = Vehicle CG.

OUT(20) "Fd_LF" (N), LF damping force, gen-name = Damping force, rigibody name = LF damper.

OUT(21) "Fd_LR" (N), LR damping force, gen-name = Damping force, rigibody name = LR damper.

OUT(22) "Fd_RF" (N), RF damping force, gen-name = Damping force, rigibody name = RF damper.

OUT(23) "Fd_RR" (N), RR damping force, gen-name = Damping force, rigibody name = RR damper.

OUT(24) "Fs_LF" (N), LF spring force, gen-name = Spring force, rigibody name = LF spring.

OUT(25) "Fs_LR" (N), LR spring force, gen-name = Spring force, rigibody name = LR spring.

OUT(26) "Fs_RF" (N), RF spring force, gen-name = Spring force, rigibody name = RF spring.

OUT(27) "Fs_RR" (N), RR spring force, gen-name = Spring force, rigibody name = RR spring.

OUT(28) "Fx_LF" (N), LF tire longitudinal force, gen-name = Tire longitudinal force, rigibody name = LF tire.

OUT(29) "Fx_LR" (N), LR tire longitudinal force, gen-name = Tire longitudinal force, rigibody name = LR tire.

OUT(30) "Fx_RF" (N), RF tire longitudinal force, gen-name = Tire longitudinal force, rigibody name = RF tire.

OUT(31) "Fx_RR" (N), RR tire longitudinal force, gen-name = Tire longitudinal force, rigibody name = RR tire.

OUT(32) "Fy_LF" (N), LF tire lateral force, gen-name = Tire lateral force, rigibody name = LF tire.

OUT(33) "Fy_LR" (N), LR tire lateral force, gen-name = Tire lateral force, rigibody name = LR tire.

OUT(34) "Fy_RF" (N), RF tire lateral force, gen-name = Tire lateral force, rigibody name = RF tire.

OUT(35) "Fy_RR" (N), RR tire lateral force, gen-name = tire lateral force, rigibody name = RR tire.

OUT(36) "Fz_LF" (N), LF tire vertical load, gen-name = Tire vertical load, rigibody name = LF tire.

OUT(37) "Fz_LR" (N), LR tire vertical load, gen-name = Tire vertical load, rigibody name = LR tire.

OUT(38) "Fz_RF" (N), RF tire vertical load, gen-name = Tire vertical load, rigibody name = RF tire.

OUT(39) "Fz_RR" (N), RR tire vertical load, gen-name = Tire vertical load, rigibody name = RR tire.

OUT(40) "JounceLF" (mm), LF susp. comp. deflection, gen-name = Jounce, rigibody name = LF suspension.

OUT(41) "JounceLR" (mm), LR susp. comp. deflection, gen-name = Jounce, rigibody name = LR suspension.

OUT(42) "JounceRF" (mm), RF susp. comp. deflection, gen-name = Jounce, rigibody name = RF suspension.

OUT(43) "JounceRR" (mm), RR susp. comp. deflection, gen-name = Jounce, rigibody name = RR suspension.

OUT(44) "Kappa_LF" (-), LF tire longitudinal slip, gen-name = Longitudinal slip, rigibody name = LF tire.

OUT(45) "Kappa_LR" (-), LR tire longitudinal slip, gen-name = Longitudinal slip, rigibody name = LR tire.

OUT(46) "Kappa_RF" (-), RF tire longitudinal slip, gen-name = Longitudinal slip, rigibody name = RF tire.

OUT(47) "Kappa_RR" (-), RR tire longitudinal slip, gen-name = Longitudinal slip, rigibody name = RR tire.

OUT(48) "Mybrk_LF" (N-m), LF brake torque, gen-name = Brake torque, rigibody name = LF Wheel.

OUT(49) "Mybrk_LR" (N-m), LR brake torque, gen-name = Brake torque, rigibody name = LR Wheel.

OUT(50) "Mybrk_RF" (N-m), RF brake torque, gen-name = Brake torque, rigibody name = RF Wheel.

OUT(51) "Mybrk_RR" (N-m), RR brake torque, gen-name = Brake torque, rigibody name = RR Wheel.

OUT(52) "Mydrv_LF" (N-m), LF drive torque, gen-name = Drive torque, rigibody name = LF Wheel.

OUT(53) "Mydrv_LR" (N-m), LR drive torque, gen-name = Drive torque, rigibody name = LR Wheel.

OUT(54) "Mydrv_RF" (N-m), RF drive torque, gen-name = Drive torque, rigibody name = RF Wheel.

OUT(55) "Mydrv_RR" (N-m), RR drive torque, gen-name = Drive torque, rigibody name = RR Wheel.

OUT(56) "Mz_LF" (N-m), LF tire aligning moment, gen-name = Tire aligning moment, rigibody name = LF tire.

OUT(57) "Mz_LR" (N-m), LR ire aligning moment, gen-name = Tire aligning moment, rigibody name = LR tire.

OUT(58) "Mz_RF" (N-m), RF tire aligning moment, gen-name = Tire aligning moment, rigibody name = RF tire.

OUT(59) "Mz_RR" (N-m), RR tire aligning moment, gen-name = Tire aligning moment, rigibody name = RR tire.

OUT(60) "Pbrk_con" (MPa), Brake control input, gen-name = Line pressure, rigibody name = Control.

OUT(61) "Pitch" (deg), Body pitch, gen-name = Pitch, rigibody name = Body.

OUT(62) "Roll" (deg), Body roll, gen-name = Roll, rigibody name = Body.

OUT(63) "Roll_WLF" (deg), LF wheel inclination angle, gen-name = Inclination angle, rigibody name = LF wheel.

OUT(64) "Roll_WLR" (deg), LR wheel inclination angle, gen-name = Inclination angle, rigibody name = LR wheel.

OUT(65) "Roll_WRF" (deg), RF wheel inclination angle, gen-name = Inclination angle, rigibody name = RF wheel.

OUT(66) "Roll_WRR" (deg), RR wheel inclination angle, gen-name = Inclination angle, rigibody name = RR wheel.

OUT(67) "Rot_WLF" (rev), Rotation of LF wheel, gen-name = Rotation of wheel, rigibody name = LF wheel.

OUT(68) "Rot_WLR" (rev), Rotation of LR wheel, gen-name = Rotation of wheel, rigibody name = LR wheel.

OUT(69) "Rot_WRF" (rev), Rotation of RF wheel, gen-name = Rotation of wheel, rigibody name = RF wheel.

OUT(70) "Rot_WRR" (rev), Rotation of RR wheel, gen-name = Rotation of wheel, rigibody name = RR wheel.

OUT(71) "Steer_LF" (deg), LF road wheel steer angle, gen-name = Steer angle (at road), rigibody name = LF wheel.

OUT(72) "Steer_LR" (deg), LR road wheel steer angle, gen-name = Steer angle (at road), rigibody name = LR wheel.

OUT(73) "Steer_RF" (deg), RF road wheel steer angle, gen-name = Steer angle (at road), rigibody name = RF wheel.

OUT(74) "Steer_RR" (deg), RR road wheel steer angle, gen-name = Steer angle (at road), rigibody name = RR wheel.

OUT(75) "Steer_SW" (deg), Steering-wheel angle, gen-name = Steering-wheel angle, rigibody name = Control.

OUT(76) "Throttle" (-), Throttle input, gen-name = Input, rigibody name = Control.

OUT(77) "Vx" (kph), Vehicle longitudinal velocity, gen-name = Longitudinal velocity, rigibody name = Vehicle CG.

OUT(78) "Vx_IErr" (m), Integrated velocity error, gen-name = Translation, rigibody name = Control.

OUT(79) "Vx_LF" (kph), LF wheel rolling speed, gen-name = Effective rolling speed, rigibody name = LF wheel.

OUT(80) "Vx_LR" (kph), LR wheel rolling speed, gen-name = Effective rolling speed, rigibody name = LR wheel.

OUT(81) "Vx_RF" (kph), RF wheel rolling speed, gen-name = Effective rolling speed, rigibody name = RF wheel.

OUT(82) "Vx_RR" (kph), RR wheel rolling speed, gen-name = Effective rolling speed, rigibody name = RR wheel.

OUT(83) "Vy" (kph), Vehicle lateral velocity, gen-name = Lateral velocity, rigibody name = Vehicle CG.

OUT(84) "Vz" (kph), Vehicle vertical velocity, gen-name = Vertical velocity, rigibody name = Vehicle CG.

OUT(85) "Xo" (m), X position of car origin, gen-name = X position, rigibody name = Car body.

OUT(86) "X_CG" (m), X position of vehicle CG, gen-name = X position, rigibody name = Vehicle CG.

OUT(87) "X_WLF" (m), X coordinate of LF wheel center, gen-name = X coordinate of wheel center, rigibody name = LF wheel.

OUT(88) "X_WLR" (m), X coordinate of LR wheel center, gen-name = X coordinate of wheel center, rigibody name = LR wheel.

OUT(89) "X_WRF" (m), X coordinate of RF wheel center, gen-name = X coordinate of wheel center, rigibody name = RF wheel.

OUT(90) "X_WRR" (m), X coordinate of RR wheel center, gen-name = X coordinate of wheel center, rigibody name = RR wheel.

OUT(91) "Yaw" (deg), Vehicle yaw, gen-name = Yaw, rigibody name = Vehicle.

OUT(92) "Yaw_WLF" (deg), Yaw angle of LF wheel, gen-name = Yaw angle of wheel, rigibody name = LF wheel.

OUT(93) "Yaw_WLR" (deg), Yaw angle of LR wheel, gen-name = Yaw angle of wheel, rigibody name = LR wheel.

OUT(94) "Yaw_WRF" (deg), Yaw angle of RF wheel, gen-name = Yaw angle of wheel, rigibody name = RF wheel.

OUT(95) "Yaw_WRR" (deg), Yaw angle of RR wheel, gen-name = Yaw angle of wheel, rigibody name = RR wheel.

OUT(96) "Yo" (m), Y position of car origin, gen-name = Y position, rigibody name = Car body.

OUT(97) "Y_CG" (m), Y position of vehicle CG, gen-name = Y position, rigibody name = Vehicle CG.

OUT(98) "Y_WLF" (m), Y coordinate of LF wheel center, gen-name = Y coordinate of wheel center, rigibody name = LF wheel.

OUT(99) "Y_WLR" (m), Y coordinate of LR wheel center, gen-name = Y coordinate of wheel center, rigibody name = LR wheel.

OUT(100) "Y_WRF" (m), Y coordinate of RF wheel center, gen-name = Y coordinate of wheel center, rigibody name = RF wheel.

OUT(101) "Y_WRR" (m), Y coordinate of RR wheel center, gen-name = Y coordinate of wheel center, rigibody name = RR wheel.

OUT(102) "Zo" (m), Z position of car origin, gen-name = Z position, rigibody name = Car body.

OUT(103) "Z_CG" (m), Z position of vehicle CG, gen-name = Z position, rigibody name = Vehicle CG.

OUT(104) "Z_WLF" (m), Z coordinate of LF wheel center, gen-name = Z coordinate of wheel center, rigibody name = LF wheel.

OUT(105) "Z_WLR" (m), Z coordinate of LR wheel center, gen-name = Z coordinate of wheel center, rigibody name = LR wheel.

OUT(106) "Z_WRF" (m), Z coordinate of RF wheel center, gen-name = Z coordinate of wheel center, rigibody name = RF wheel.

OUT(107) "Z_WRR" (m), Z coordinate of RR wheel center, gen-name = Z coordinate of wheel center, rigibody name = RR wheel.

Creating Lists of Outputs

A list of output variables can be created from within the software. This list is a subset of the information contained in the README files. If you happen to use a version of CarSimEd that has been customized, you might want to make the list for reference purposes. To make a list:

1. Go to the Runs library, and find a run involving the vehicle of interest.
2. If the run has not already been made, click the **Run** button.
3. Click the **Plot** button to bring up WinEP.

4. Select the menu item **Show Channel List** from the **View** menu.
5. A list of all channels will appear. In addition to the list of channels, the window has two buttons: Cancel and Save. To save a listing to a file, click the Save button. The Windows file browser will then appear.
6. In the Windows file browseer, enter a name for the text file and save it in a directory where you can find it later. The default name is `Chans.txt`.
7. Click the **Save** button to create the file.
8. You will be returned to the WinEP graphics window.
9. Exit WinEP.